

On adaptively secure prefix encryption under LWE

Giorgos Zirdelis*

March 21, 2023

Abstract

Prefix Encryption is a public-key encryption scheme, where ciphertexts are associated with a string y and secret keys are associated with a string x . Any secret key for which x is a prefix of y , allows to decrypt a ciphertext associated with x . Secret keys are issued by a trusted authority, that additionally publishes a set of public parameters used for encryption and decryption.

Prefix encryption was formalized in a work of Lewko and Waters (EUROCRYPT'14) to show that *certain* partitioning proof techniques fail to achieve adaptive security without incurring an exponential loss in the security reduction. An exponential loss requires a stronger security assumption to achieve adaptive security, which is undesirable. Prefix encryption can be constructed from Hierarchical Identity-Based Encryption (HIBE) or Attribute-Based Encryption (ABE), which implies that the same partitioning techniques must incur an exponential loss in the security reduction when applied to HIBE and ABE. While it remains a long-standing open problem to achieve adaptive security with a polynomial reduction loss for HIBE or ABE under LWE, the same work showed how to obtain adaptively secure prefix encryption from adaptively secure Identity-Based Encryption.

In this work, we give a construction of an adaptively secure prefix encryption scheme with a polynomial reduction loss, *directly* from LWE. To encrypt to a string y we derive a public key for every prefix of y from a fixed set of public parameters using lattice-based homomorphic operations, similar to previous work. Our approach differs in the secret key generation, where a secret key for x takes into account and ties together *every* prefix of x , and our techniques may be of independent interest. This leaves open the possibility for the secret keys to be extended in a way that could lead to adaptive security of delegation functionalities, in the future. For security, we leverage a work of Tsabary (CRYPTO'19) and extend it to obtain our result.

*Modulus Labs. Email: zirdelis.g@northeastern.edu

A preliminary version of this paper appears in the proceedings of CSCML 2023. This is the full version. Work done while the author was at Northeastern University and the University of Maryland.

1 Introduction

Prefix Encryption (PRE) is a public-key primitive [LW14], that allows to encrypt a message m with respect to a string y , using set of public parameters generated by a trusted authority. Anyone can request from the authority a secret key that is associated with a string x , and can then decrypt any ciphertext for which x is a prefix of y . Moreover, we do not require any hiding properties for strings x and y . Such a mechanism allows for a fine-grained access control of encrypted data, when compared to the all-or-nothing approach of traditional public-key encryption, and thus leads to more advanced applications.

An application of prefix encryption. To highlight the usefulness of prefix encryption, we describe an application for networks but emphasize that it is a primitive which could have a broader scope for applications.

Consider a *trusted* network router that is responsible for forwarding packets encrypted using a prefix encryption scheme, to large set of clients behind a subnetwork. Each packet is encrypted to a string that matches the network address of a client, and each client has a secret key that decrypts only packets associated with its network address. The router using a *single* prefix encryption secret key that matches the prefix of the subnetwork address for all clients, is able to decrypt and thoroughly inspect for malware any packets before these are delivered to the clients.

While large subnetworks could facilitate a *super-polynomial* network address range despite serving polynomially many clients, prefix encryption makes this possible using a *single* secret key for such a range without explicitly requiring the secret key for every client, especially if we consider that a client's network address might change over time.

The above example could be generalized to more complex networks with multiple routers and subnetworks, and thus prefix encryption can deliver an advanced packet inspection mechanism.

Prior work on prefix encryption. Prefix encryption was formally defined in [LW14] to show that *certain* partitioning proof techniques fail to achieve adaptive security without incurring an exponential loss in the security reduction. Prefix encryption can be constructed from Hierarchical Identity-Based Encryption (HIBE) [HL02, GS02] or Attribute-Based Encryption (ABE) [SW05, GPSW06]. Consequently, the same partitioning proof techniques will also incur an exponential loss in the reduction when applied to HIBE or ABE. Moreover, the work of [LW14] showed that specifically for prefix encryption their impossibility result can be bypassed using Identity-Based Encryption (IBE) [Sha84, BF01]. We review the [LW14] construction in Section 1.2.1 before presenting our scheme. In comparison, we give a *direct* construction of prefix encryption from LWE, and our techniques may be of independent interest. Finally, the work of [WTSB16] utilizes prefix encryption in the context of private service discovery.

Security for prefix encryption. Security for prefix encryption guarantees that any adversary in possession of secret keys that are not authorized to decrypt a ciphertext ct , should not learn anything about the encrypted message. The *adaptive* security model captures exactly this *realistic* security notion, and thus is desirable to obtain.

In the adaptive security game for prefix encryption, the adversary first receives the public parameters from the challenger. The adversary then adaptively requests secret keys sk_x for any string x of its choosing. After this step, the adversary sends a challenge string y^* to the challenger, that creates a ciphertext ct_{y^*} encrypting a random bit, and sends ct_{y^*} to the adversary. Finally, the adversary adaptively requests again secret keys sk_x , and then outputs its guess on for the encrypted

bit in ct_{y^*} . We restrict the adversary to only request secret keys sk_x for which x is not a prefix of y^* , otherwise it would trivially win the security game. The scheme is secure if the advantage of the adversary is $1/2 + \text{negl}(\lambda)$ where λ is a security parameter. For adaptive PRE we require a polynomial reduction loss in the (maximum) length $|y|$ of the string y that is associated with any ciphertext. Any string y is assumed to be bounded by a polynomial in the security parameter. Thus, a polynomial *reduction* loss in $|y|$ implies an overall negligible loss in the security parameter.

The *selective* security model [CHK03] differs from the above at the challenge step. The adversary must now declare its challenge y^* *before* it receives the public parameters. Selective security can be upgraded to adaptive in a generic way using *complexity leveraging* [BB04], but this inherently requires an exponential loss in the security reduction. A simple strategy is to guess the challenge $y^* \in \{0, 1\}^*$. This succeeds with probability $2^{-|y|}$ and implies an exponential reduction loss. In turn, this implies a larger value for the security parameter to account for this loss, and thus a stronger security assumption which is undesirable.

On lattice-based HIBE and ABE. It remains a long-standing open problem to construct lattice-based adaptively secure HIBE and ABE with a polynomial reduction loss.

Lattice-based HIBE constructions [CHKP10, ABB10a, ABB10b, DG17b, DG17a, DGHM18] are proven to be selectively secure. We note that delegatable PRFs [KPTZ13, BW13, BGI14] can be used in the selective HIBE from [DG17b, DG17a], with essentially the same reduction loss as the adaptive delegatable PRF loss [FKPR14, HKKW19]. Currently, there is no adaptive delegatable PRF with a polynomial reduction loss in the standard model. Similarly, lattice-based ABE for circuits and beyond [GVW13, BGG⁺14, GVW15, AFS19] is proven secure in the selective model. Furthermore, the works of [BV16, GKW16] proved ABE security in the semi-adaptive setting [CW14b]. For semi-adaptive security, the adversary must declare its challenge after it receives the public parameters but *before* it makes any secret key queries. Specifically the work of [GKW16] showed a transformation of selective Functional Encryption (FE) [SW05, BW07, KSW08, LOS⁺10, O’N10, BSW11] to semi-adaptive FE (and thus applies to ABE) under the assumption that public-key encryption exists, alluding that there might be a bigger gap between semi-adaptive and adaptive security than there is between selective and semi-adaptive security. For the powerful notion of FE though, the works of [Wat15, ABSV15] showed how to upgrade selective FE to adaptive, but this is not known for ABE. Note that for HIBE the exponential reduction loss is incurred in the hierarchy depth, and for key-policy ABE in the attribute length. Both these quantities can be assumed to be bounded by a polynomial in the security parameter, and thus the overall reduction loss is exponential in the security parameter. Furthermore, HIBE and ABE are seemingly incomparable due to the key delegation mechanism required for HIBE. We review the notion of ABE in Section 1.2.2.

Finally, the work of Tsabary [Tsa19] makes partial progress in adaptive ABE security by constructing adaptively secure ABE for the class of t -CNF predicates for constant t , along with an new adaptive IBE scheme, by building partially on [DKNY18]. See [DKN⁺20] for an updated version of [DKNY18], and [KNYY20] for a follow-up work on adaptive inner-product encryption.

We note that adaptively secure HIBE and ABE for NC^1 with a polynomial reduction loss, exist under (standard) group assumptions (e.g. [GH09, Wat09, LW10, LW11, Lew12, CW13, CW14a] and [LOS⁺10, OT10, LW12, CGW15, KL15, KW19]), and adaptive ABE for general circuits exists under the powerful notion of functional encryption [Wat15, ABSV15, JLS21], but these constructions rely on (well-founded) assumptions that are broken in the presence of quantum computers.

Our results. In this work, we present an adaptively secure prefix encryption scheme *directly* from LWE with a polynomial reduction loss in the length of the challenge string.

Theorem 1.1 (Informal). There exists an adaptively secure prefix encryption scheme for a-priori bounded length strings, under the polynomial hardness of approximating worst-case lattice problems to within sub-exponential factors.

1.1 Paper organization

In Section 1.2 we give a detailed overview of our Prefix Encryption (PRE) scheme, and review relevant work. In Section 1.3 we discuss open problems relevant to our work. In Section 2 we review all necessary technical background. In Section 3 we give the PRE definition and security model. In Section 4 we give the full construction and security proof of our PRE scheme.

1.2 Technical overview

In this section, we give the technical details of our PRE construction. In Section 1.2.1 we describe the construction of PRE from IBE. In Section 1.2.2 we review in detail the related work for ABE and IBE that we require in our construction. In Section 1.2.3 we give the details of our PRE scheme. For this overview we define $[\ell] = \{1, \dots, \ell\}$ and denote with λ the security parameter.

1.2.1 PRE from IBE

The PRE construction from adaptive IBE, explicitly written in [LW14], works as follows: to encrypt a message m to a string $y \in \{0, 1\}^\ell$ the encryptor creates ℓ IBE ciphertexts for m , one for each prefix of y serving as the identity. Note that identities of varying length can be compressed to λ bits using standard hashing techniques. A secret key for x corresponds to an IBE secret key $\mathbf{sk}_x$ for identity x . To decrypt $\mathbf{ct}_y$ using $\mathbf{sk}_x$, we select the IBE ciphertext for which x matches some prefix of y , if any, and then apply IBE decryption using $\mathbf{sk}_x$. Adaptive PRE security follows readily from adaptive IBE security. In our scheme, the secret keys grow linearly with the prefix length when compared to the IBE construction, but in contrast our construction leaves open the possibility of secret key delegation, as required for example by HIBE which was the original motivation for this work. Note that HIBE secret key delegation is not known to follow by IBE in a black-box manner.

1.2.2 Related work

We review the ABE from [BGG⁺14] that is used in the adaptive IBE from [Tsa19] that we review in detail next, and on which we rely and extend to obtain our scheme.

ABE and matrix homomorphisms. In an ABE scheme, a trusted authority generates a set of public parameters $\mathbf{pp}$ and a master secret key $\mathbf{msk}$. Anyone can use $\mathbf{pp}$ to create a ciphertext $\mathbf{ct}_x$ that encrypts a message $m \in \{0, 1\}$ under an attribute $x \in \{0, 1\}^\ell$. The ciphertext $\mathbf{ct}_x$ does not hide the attribute x but only the message m .

Given a (public) boolean circuit $f : \{0, 1\}^\ell \rightarrow \{0, 1\}$, anyone can homomorphically evaluate f on $\mathbf{ct}_x$ and obtain $\mathbf{ct}_{f(x)}$ since x is not hidden. The authority uses $\mathbf{msk}$ to issue a secret key $\mathbf{sk}_f$ for a circuit f that is combined with $\mathbf{ct}_{f(x)}$ to recover m if and only if $f(x) = 0$ ¹, otherwise m remains hidden. Security for ABE requires that an adversary can request secret keys $\mathbf{sk}_{f_i}$ for any circuit f_i , and cannot recover a message m that is encrypted under a challenge attribute x^* of the adversary's

¹For technical reasons we use the inverted condition of $f(x) = 0$ to authorize decryption, instead of the more common $f(x) = 1$.

choosing, given that for all requested secret keys sk_{f_i} it holds $f_i(x^*) = 1$ (i.e. decryption is not authorized).

At a high level, the scheme in [BGG⁺14] constructs ABE for circuits in the following way. The public parameters contain matrices $\mathbf{B}_1, \dots, \mathbf{B}_\ell$. We encode an attribute $x \in \{0, 1\}^\ell$ as $\mathbf{C}_i = \mathbf{B}_i + x_i \cdot \mathbf{G}$ where $\mathbf{G}$ is a fixed public "gadget" matrix. We encrypt under LWE with $\mathbf{C}_i$ serving as public keys and obtain a ciphertext ct_x . Anyone can homomorphically compute a circuit f on ct_x to obtain a new LWE ciphertext $\text{ct}_{f(x)}$ with $\mathbf{B}_f + f(x) \cdot \mathbf{G}$ as the public key. The secret key sk_f the authority provides satisfies the equation $\mathbf{B}_f \cdot \text{sk}_f = \mathbf{0}$. Thus, when $f(x) = 0$ we can decrypt. Notice that the authority does not know x , but is able to issue keys because the matrix $\mathbf{B}_f$ depends *only* on $\mathbf{B}_i$.

Furthermore, the scheme supports circuit composition. After we evaluate f to obtain $\mathbf{C}_f = \mathbf{B}_f + f(x) \cdot \mathbf{G}$ (assume $f(x)$ is multi-bit output) and then encrypt, a third party interprets $f(x)$ as an attribute and can homomorphically evaluate a circuit g on $f(x)$ to obtain a ciphertext for $g \circ f$,

$$\mathbf{C}_{g \circ f} = \mathbf{B}_{g \circ f} + g(f(x)) \cdot \mathbf{G}$$

In particular, we can first encode x and then apply a function f on $\mathbf{C}_i$ to obtain $\mathbf{C}_f = \mathbf{B}_f + f(x) \cdot \mathbf{G}$ before we encrypt with $\mathbf{C}_f$ as the public key. This can be interpreted as a pre-processing step. Importantly, $\mathbf{B}_f$ does not depend on x . In our scheme, f will be a pseudorandom function (PRF) with hardcoded input and x will be a PRF key. Thus, $\mathbf{B}_f$ and $f(x)$ reveal nothing about the PRF key x . In this case, we encrypt using $\mathbf{C}_f$ as the LWE public key and $f(x)$ as the resulting (new) attribute, where f is a multi-bit output PRF circuit.

The [Tsa19] IBE. We give a detailed review of the adaptive IBE from [Tsa19] that contains a core idea which we adapt in our scheme. In the process, we introduce notation that we use heavily in the rest of the section.

In IBE, ciphertexts are associated with a identity string y , and secret keys sk_x with a identity string x . Decryption succeeds if and only if $y = x$. For security, the adversary must declare a challenge identity y^* that must be different from any identity x for which it has requested a secret key sk_x . The adversary first receives the public parameters, and then requests secret keys adaptively before and after it announces y^* .

The core idea from the [Tsa19] IBE. At a high level the scheme works as follows: to encrypt to identity x , the encryptor samples a PRF key k and uses the [BGG⁺14] ABE to obtain a PRF output r_x for PRF input (k, x) . The value r_x is the attribute in the ABE, and the identity x is tied to the ciphertext by properties of the ABE [BGG⁺14]. The authority uses its own PRF key k^* to obtain a PRF output r_x^* for input (k^*, x) . It outputs a secret key for x that decrypts if $r_x \neq r_x^*$ which happens with high probability over the choice of k, k^* and correctness follows. In general, we denote with r_z^* PRF values using key k^* and with r_z PRF values using key k .

In the security reduction (and the real system) the secret key for any identity x only reveals r_x^* that is obtained using k^* . Moreover, the scheme public parameters are set up in a way such that the challenge ciphertext ct_{y^*} reveals the value $r_{y^*}^*$ and nothing about k^* . In addition, the value $r_{y^*}^*$ does not appear in any requested secret key sk_x (for which we use k^*) since the challenge identity y^* must be different from any requested identity x . This allows for an adaptive reduction under (adaptive) PRF security and ABE techniques from [BGG⁺14].

The IBE in [Tsa19] relies on the properties of the ABE in [BGG⁺14] which is known to be selectively secure for general policies (i.e. boolean circuits). Specifically for the IBE policy, [Tsa19] uses

a PRF in the pre-processing step and obtains adaptive IBE from the selective ABE in [BGG⁺14]. Below we give a slightly different presentation of the original scheme in [Tsa19, Section 1.2].²

IBE circuits. We first define three circuits used in our IBE overview, following closely [Tsa19]. For a PRF $f : \{0, 1\}^{\ell_k} \times \{0, 1\}^{\ell_{\text{id}}} \rightarrow \{0, 1\}^{\ell_{\text{out}}}$ and every $y \in \{0, 1\}^{\ell_{\text{id}}}$, define f_y to be the circuit with hardcoded PRF input y . The circuit f_y takes as input the PRF key k and outputs $f_y(k) = f(k, y)$.

For every $r^* \in \{0, 1\}^{\ell_{\text{out}}}$ define the *equality test* circuit $g_{r^*} : \{0, 1\}^{\ell_{\text{out}}} \rightarrow \{0, 1\}$ as $g_{r^*}(r) = 1$ if $r = r^*$, and 0 otherwise. Given r^* , we can construct g_{r^*} on the fly.

Finally, we define circuit composition in the natural way: $(g_r \circ f_y)(\cdot) = g_r(f_y(\cdot))$.

IBE construction. We review the full IBE construction, starting with the public parameters pp and master secret key msk . For a fixed public "gadget" matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$, the authority samples:

$$\text{pp} = \{\mathbf{u}, \mathbf{A}, \mathbf{B}_1, \dots, \mathbf{B}_{\ell_k}\} \quad \text{and} \quad \text{msk} = \{\mathbf{T}_{\mathbf{A}}, k^*\}$$

with $\mathbf{u} \in \mathbb{Z}_q^n$, $\mathbf{A}, \mathbf{B}_i \in \mathbb{Z}_q^{n \times m}$, $\mathbf{T}_{\mathbf{A}} \in \mathbb{Z}^{m \times m}$ with low-norm entries, and $k^* \in \{0, 1\}^{\ell_k}$ is a random PRF key. For any $\mathbf{F} \in \mathbb{Z}_q^{n \times m}$, the matrix $\mathbf{T}_{\mathbf{A}}$ allows to sample a low-norm matrix $\mathbf{R}_{\mathbf{F}}$ such that $[\mathbf{A} \parallel \mathbf{F}] \cdot \mathbf{R}_{\mathbf{F}} = \mathbf{u}$. We implicitly assume a public circuit description of a PRF f as defined above.

To encrypt a message $\text{msg} \in \{0, 1\}$ to identity $y \in \{0, 1\}^{\ell_{\text{id}}}$, first sample a fresh PRF key $k \in \{0, 1\}^{\ell_k}$ and set $r_y = f(k, y)$. Then, derive the circuit f_y , and use $\{\mathbf{B}_i\}$ from pp to homomorphically compute the matrix $\mathbf{B}_{f_y}$ that depends only on f_y , and thus "ties" y to $\mathbf{B}_{f_y}$.

Finally, output the ciphertext ct_y (we abuse notation and represent many bits with $r_y \cdot \mathbf{G}$):

$$\text{ct}_y = (y, r_y, \mathbf{s}^\top \mathbf{A} + \mathbf{e}_a^\top, \mathbf{s}^\top (\mathbf{B}_{f_y} + r_y \cdot \mathbf{G}) + \mathbf{e}^\top, \mathbf{s}^\top \mathbf{u} + e_{\text{msg}} + \text{msg} \lceil q/2 \rceil)$$

where $\mathbf{s}$ is random and the error terms are of low norm.

To sample a secret key sk_x the authority first uses msk to set $r_x^* = f(k^*, x)$, and then derives the circuit $C_x = g_{r_x^*} \circ f_x$. Then it uses $\{\mathbf{B}_i\}$ and C_x to homomorphically compute $\mathbf{B}_{C_x}$. Finally, it uses $\mathbf{T}_{\mathbf{A}}$ to sample a low-norm $\mathbf{R}_{\text{id}}$ such that

$$[\mathbf{A} \parallel \mathbf{B}_{C_x}] \cdot \mathbf{R}_x = \mathbf{u}$$

and outputs $\text{sk}_x = (x, r_x^*, \mathbf{R}_x)$.

For sk_x and ct_x , i.e. when $x = y$, the decryptor uses r_x^* to homomorphically compute $g_{r_x^*}$ using ct_x since r_x is public, and obtains:

$$\text{ct}'_x = (x, r_x, \mathbf{s}^\top \mathbf{A} + \mathbf{e}_a^\top, \mathbf{s}^\top (\mathbf{B}_{C_x} + g_{r_x^*}(r_x) \cdot \mathbf{G}) + \mathbf{e}_d^\top, \mathbf{s}^\top \mathbf{u} + e_m + m \lceil q/2 \rceil).$$

Correctness holds with overwhelming probability over the choice of k, k^* which implies that $r_x \neq r_x^*$. Thus $g_{r_x^*}(r_x) = 0$, and if we multiply $(\mathbf{s}^\top \mathbf{A} + \mathbf{e}_a^\top \parallel \mathbf{s}^\top \mathbf{B}_{C_x} + \mathbf{e}_d^\top)$ with $\mathbf{R}_x$ we obtain $\mathbf{s}^\top \mathbf{u} + \tilde{e}$ and we can recover m .

²[Tsa19, Section 1.2] for IBE uses the circuit $\bar{I}_r(r') = 1$, if $r' \neq r$ and 0 otherwise. We use $I_r(r') = 1$, if $r' = r$ and 0 otherwise.

IBE security proof. In the proof, the public parameters are sampled without a "trapdoor" $\mathbf{T}_\mathbf{A}$ for $\mathbf{A}$, and with $\mathbf{B}_i = \mathbf{A}\mathbf{R}_i - k_i^* \cdot \mathbf{G}$ where $\mathbf{R}_i$ is of low-norm, and k_i^* is the i -th bit of k^* :

$$\text{pp}_{\text{sim}} = \{\mathbf{u}, \mathbf{A}, \mathbf{A}\mathbf{R}_1 - k_1^* \cdot \mathbf{G}, \dots, \mathbf{A}\mathbf{R}_{\ell_k} - k_{\ell_k}^* \cdot \mathbf{G}\} \quad \text{and} \quad \text{msk}_{\text{sim}} = \{\mathbf{R}_1, \dots, \mathbf{R}_{\ell_k}, k^*\}$$

Using standard arguments [BGG⁺14, BV16], we get the *statistical* indistinguishability, $\text{pp} \stackrel{s}{\approx} \text{pp}_{\text{sim}}$. For a secret key sk_x , the simulator first derives C_x and $\mathbf{B}_{C_x}$. It then uses $\mathbf{A}, \{\mathbf{R}_i\}$ to compute $\mathbf{R}_{C_x}$ such that,

$$\mathbf{B}_{C_x} = \mathbf{A}\mathbf{R}_{C_x} - g_{r_x^*}(r_x^*) \cdot \mathbf{G} = \mathbf{A}\mathbf{R}_{C_x} - \mathbf{G}$$

Finally, it uses $\mathbf{R}_{C_x}$ and $\mathbf{G}$ to sample a low-norm $\mathbf{R}_x$ such that $[\mathbf{A} \parallel \mathbf{B}_{C_x}] \cdot \mathbf{R}_x = \mathbf{u}$ while maintaining indistinguishability, under appropriate parameters, from the real system that uses $\mathbf{T}_\mathbf{A}$ to sample $\mathbf{R}_x$.

To encrypt a random bit to a challenge identity y^* , the simulator "samples" a PRF key k with $k = k^*$. It sets $r_{y^*}^* = f(k^*, y)$ and computes the public key for ct_{y^*} (we abuse notation and represent many bits with $r_{y^*}^* \cdot \mathbf{G}$):

$$\mathbf{B}_{f_{y^*}} + r_{y^*}^* \cdot \mathbf{G} = (\mathbf{A}\mathbf{R}_{f_{y^*}} - r_{y^*}^* \cdot \mathbf{G}) + (r_{y^*}^*) \cdot \mathbf{G} = \mathbf{A}\mathbf{R}_{f_{y^*}}$$

This results in a public key for the challenge ciphertext ct_{y^*} that does not contain any " $\mathbf{G}$ -trapdoor" and ct_{y^*} is proven [BGG⁺14, Tsa19] to be indistinguishable from the uniform distribution under LWE. Note that the adversary is not allowed to request a secret key for y^* . Thus, the value $r_{y^*}^*$ remains pseudorandom by (adaptive) PRF security, even given the PRF values in sk_x . This implies that the adversary cannot distinguish the real game from the simulated one with more than a negligible advantage.

1.2.3 Our PRE construction

In this section, we describe our adaptive PRE scheme. We set the PRE string alphabet to $\{0, 1\}$ and let strings to be vectors in $(\{0, 1\})^{\ell_{\text{str}}}$ where ℓ_{str} is an a-priori fixed bound. To simplify notation, we write $y \in \{0, 1\}^{\ell_{\text{str}}}$ instead of $y \in (\{0, 1\})^{\ell_{\text{str}}}$. We first define prefix sets that allow us to associate prefix strings to 2λ -bit strings using a collision-resistant hash function (CRHF) and then proceed with our scheme overview.

Prefix sets and string encodings. Our first observation, similar to [LW14], is that any string $y = (y_1, \dots, y_\ell) \in \{0, 1\}^\ell$, i.e. $y_j \in \{0, 1\}$, can be expanded to a set that contains all the ℓ (non-empty) prefix strings of x :

$$S_y = \{ (y_1), (y_1, y_2), \dots, (y_1, \dots, y_i), \dots, (y_1, \dots, y_i, \dots, y_\ell) \}.$$

We rely on standard techniques to encode every element of S_y into λ bits using a collision-resistant hash function (CRHF). For example, we can first apply an Error-Correcting Code to " $(y_1, \dots, y_i) \parallel i$ " and then rely on the Merkle–Damgård transform [Dam90, Mer90]. That is, we assume an efficient (public) procedure $\text{Encode}(y, i)$ that outputs an encoding of *exactly* 2λ bits for the i -th prefix of y . Finding different pairs (y, i) and (y', i') such that $\text{Encode}(y, i) = \text{Encode}(y', i')$ is computationally hard assuming collision-resistance. Depending on the specifics of the Encode procedure, the length i might not be required to be part of the input, yet we choose to make it part of the input to avoid any ambiguity that could arise. To simplify notation, we denote with $y(1, i)$ the output of $\text{Encode}(y, i)$ for $i \in [\ell]$. This 2λ bit string is given as input to the PRF instead of the actual prefix $(y_1, \dots, y_i)$.

PRE scheme outline. We first outline the intuition behind our scheme before the technical overview. To encrypt to a string $y \in \{0, 1\}^\ell$ we encrypt with respect to all ℓ prefixes y , essentially encrypting to the prefix set S_y . We sample a PRF key k^{enc} and compute a PRF value for *every* prefix of y , similar to the IBE in [Tsa19]. This allows an encryption to y to carry bit-by-bit its lineage, which could potentially allow for a bit-by-bit key extension as we discuss in Section 1.3.

Our scheme differs from the generic IBE construction in the key generation phase. Our msk contains two PRF keys $k^{\text{real}}, k^{\text{fake}}$. A secret key sk_x ties together all the ℓ prefixes of $x \in \{0, 1\}^\ell$, and contains ℓ PRF values for ℓ comparison circuits g , as described in the previous section. We use k^{fake} to generate the PRF comparison values for the first $(\ell - 1)$ prefixes of x , and k^{real} to generate the PRF comparison value for the ℓ -th prefix, which is x itself. The role of using two PRF keys only matters for the security proof and not for correctness. To decrypt ct_y using sk_x , we select the ciphertext components for which x is a prefix of y , if any, and proceed similar to the previous section.

In the security proof, we embed the key k^{real} in the public parameters. This way, during key generation we are able to embed a trapdoor for the last prefix of x that is x itself, using a PRF value under k^{real} . We use k^{fake} to obtain the rest of the $(\ell - 1)$ PRF values for each of the (shorter) $(\ell - 1)$ prefixes of x , but k^{fake} does not imply a trapdoor. The PRF values for y^* and all its prefixes for the challenge ciphertext ct_{y^*} are obtained using $k^{\text{enc}} = k^{\text{real}}$. While this allows to argue LWE, we also have to argue that the "sampling" $k^{\text{enc}} = k^{\text{real}}$ during ct_{y^*} goes unnoticed by the adversary. This follows because the PRF values in ct_{y^*} remain pseudorandom given the PRF values in any secret key sk_x . In more detail, notice that any sk_x contains a PRF value under k^{real} *only* for x itself, and not for any of its (shorter) prefixes. Any (shorter) prefix of x except x itself, is allowed to be in y^* , but the PRF values for these prefixes, for any secret key, are given using k^{fake} . By the PRE security game, x itself cannot equal to y^* or be any prefix of y^* . That is, the PRF values under k^{real} for ct_{y^*} and any sk_x do not intersect, and the pseudorandomness claim follows. The reduction loss is polynomial in $\lambda, \ell_{\text{str}}$, essentially because we have to account for the pseudorandomness loss that is incurred by the PRF values that are given in ct_{y^*} and every sk_x . There are at most ℓ_{str} PRF values in either ct_{y^*} or any sk_x . Assuming the adversary makes $Q = \text{poly}(\lambda)$ key queries, the reduction loss is bounded by $(Q + 1) \cdot \ell_{\text{str}} \cdot \text{Adv}^{\text{PRF}}$ where $\text{Adv}^{\text{PRF}} = \text{negl}(\lambda, \ell_{\text{str}})$ is the pseudorandomness reduction loss for one PRF value. Note that instead of using k^{fake} we could sample random value but using this PRF key simply allows to derandomize that part of the secret key generation. Details follow.

PRE circuits. In addition to the equality test circuit g from the previous section, we rely on a circuit that computes a PRF value for any given prefix of a string y . This allows us to derive public keys for *every* prefix of y . For a PRF $f : \{0, 1\}^{\ell_k} \times \{0, 1\}^{2\lambda} \rightarrow \{0, 1\}^{\ell_{\text{out}}}$ and for every $y \in \{0, 1\}^{\ell_{\text{str}}}$, define $f_{y,i}$ to be the circuit with hardcoded PRF input: $y(1, i)$. On input a key k , the circuit $f_{y,i}$ outputs $f_{y,i}(k) = f(k, y(1, i))$. For every $y \in \{0, 1\}^{\ell_{\text{str}}}$ and $i \in [\ell_{\text{str}}]$, using the Encode procedure we obtain $y(1, i) = \text{Encode}(y, i)$ with $|y(1, i)| = 2\lambda$. Thus, we could set $\ell_{\text{str}} = 2\lambda$ in the definition of f since we evaluate f on the encoding of a string y , and not the actual string y .

PRE construction. We first define the public parameters and master secret key. The authority samples:

$$\text{pp} = \{\mathbf{u}, \mathbf{A}, \mathbf{B}_1, \dots, \mathbf{B}_{\ell_k}\} \quad \text{and} \quad \text{msk} = \{\mathbf{T}_A, k^{\text{real}}, k^{\text{fake}}\}$$

with $\mathbf{u} \in \mathbb{Z}_q^n$, $\mathbf{A}, \mathbf{B}_i \in \mathbb{Z}_q^{n \times m}$, $\mathbf{T}_A \in \mathbb{Z}^{m \times m}$ with low-norm entries, and $k^{\text{real}}, k^{\text{fake}}$ are PRF keys. Implicitly, we assume that pp contain the description of a PRF function f and of an encoding

procedure Encode.

To encrypt a message $\text{msg} \in \{0, 1\}$ to a string $y \in \{0, 1\}^\ell$, first sample a PRF key k^{enc} and set $r_i^{\text{enc}} = f(k^{\text{enc}}, y(1, i))$ for $i \in [\ell]$. Then, derive the circuit $f_{y,i}$ and use $\{\mathbf{B}_i\}$ to compute the matrix $\mathbf{B}_{f_{y,i}}$ that depends only on $f_{y,i}$, and thus "ties" $y(1, i)$ to $\mathbf{B}_{f_{y,i}}$. Therefore, the sequence of ℓ matrices $(\mathbf{B}_{f_{y,i}})_{i \in [\ell]}$ is tied with y .

Finally, set $\mathbf{c}_i$ as follows (we abuse notation and denote many bits with r_i^{enc}),

$$\mathbf{c}_i = \mathbf{s}^\top (\mathbf{B}_{f_{y,i}} + r_i^{\text{enc}} \cdot \mathbf{G}) + \mathbf{e}_i^\top \quad \text{for } i \in [\ell]$$

and output the ciphertext:

$$\text{ct}_y = (y, \{r_i^{\text{enc}}\}_{i \in [\ell]}, \mathbf{s}^\top \mathbf{A} + \mathbf{e}_a^\top, \{\mathbf{c}_i\}_{i \in [\ell]}, \mathbf{s}^\top \mathbf{u} + e_{\text{msg}} + \text{msg} \lceil q/2 \rceil).$$

The ℓ components $(r_i, \mathbf{c}_i)$ in ct_y correspond to encryption with respect to the prefix set S_y .

To sample a secret key sk_x for $x \in \{0, 1\}^\ell$ using $\text{msk} = \{\mathbf{T}_\mathbf{A}, k^{\text{real}}, k^{\text{fake}}\}$, the authority derives PRF values for the first $(\ell - 1)$ prefixes of x with the key k^{fake} . For the last prefix $x(1, \ell)$ which is x itself it uses the key k^{real} ,

$$r_i = f(k^{\text{fake}}, x(1, i)) \quad \forall i \in [\ell - 1] \quad \text{and} \quad r_\ell = f(k^{\text{real}}, x(1, \ell)).$$

Then, for each prefix $i \in [\ell]$, it derives the circuit $C_i = g_{r_i} \circ f_{x,i}$, and uses $\{\mathbf{B}_i\}$ to compute $\mathbf{B}_{C_i}$. Notice that C_i contains three pieces of information: a prefix index i , a prefix $x(1, i)$, and a PRF value r_i for that prefix. Thus, the homomorphically derived public key $\mathbf{B}_{C_i}$ is "tagged" exactly with this information. Finally, it uses $\mathbf{T}_\mathbf{A}$ to sample a low-norm $\mathbf{R}_x$ such that

$$[\mathbf{A} \parallel \mathbf{B}_{C_1} \parallel \dots \parallel \mathbf{B}_{C_\ell}] \cdot \mathbf{R}_x = \mathbf{u}$$

and outputs $\text{sk}_x = (x, \{r_i\}_{i \in [\ell]}, \mathbf{R}_x)$.

The decryptor selects the ℓ ciphertext components of ct_y for which x is a prefix of y , if any, otherwise decryption is not authorized. Then, it uses $\{r_i\}_{i \in [\ell]}$ from sk_x to homomorphically compute g_{r_i} on $\mathbf{c}_i$ and derive:

$$\mathbf{c}'_i = \mathbf{s}^\top (\mathbf{B}_{C_i} + g_{r_i}(r_i^{\text{enc}}) \cdot \mathbf{G}) + \tilde{\mathbf{e}}_i^\top \quad \text{for } i \in [\ell]$$

With overwhelming probability over the choice of $k^{\text{enc}}, k^{\text{real}}, k^{\text{fake}}$ it follows that $r_i \neq r_i^{\text{enc}}$ for all $i \in [\ell]$. Thus, $g_{r_i}(r_i^{\text{enc}}) = 0$ for all $i \in [\ell]$ and the decryptor can use $\mathbf{R}_x$ to recover m .

PRE security proof. The reason for using two PRF keys is made clear in the security proof that we describe next. The public parameters are now sampled without a "trapdoor" for $\mathbf{A}$. We set $\mathbf{B}_i = \mathbf{A}\mathbf{R}_i - k_i^{\text{real}} \cdot \mathbf{G}$ where $\mathbf{R}_i$ is of low-norm, and k_i^{real} is the i -th bit of k^{real} ,

$$\text{pp}_{\text{sim}} = \{\mathbf{u}, \mathbf{A}, \mathbf{A}\mathbf{R}_1 - k_1^{\text{real}} \cdot \mathbf{G}, \dots, \mathbf{A}\mathbf{R}_{\ell_k} - k_{\ell_k}^{\text{real}} \cdot \mathbf{G}\} \quad \text{and} \quad \text{msk}_{\text{sim}} = \{\mathbf{R}_1, \dots, \mathbf{R}_{\ell_k}, k^{\text{real}}, k^{\text{fake}}\}$$

As in the previous section, we have $\text{pp} \stackrel{s}{\approx} \text{pp}_{\text{sim}}$.

To sample a secret key sk_x , the simulator derives C_i and then computes $\mathbf{B}_{C_i}$. It uses $\mathbf{A}, \{\mathbf{R}_i\}$ to compute $\mathbf{R}_{C_i}$ such that for $i \in [\ell - 1]$,

$$\mathbf{B}_{C_i} = \mathbf{A}\mathbf{R}_{C_i} - g_{r_i^{\text{fake}}}(r_i^{\text{real}}) \cdot \mathbf{G} = \mathbf{A}\mathbf{R}_{C_i}$$

and for $i = \ell$,

$$\mathbf{B}_{C_\ell} = \mathbf{A}\mathbf{R}_{C_\ell} - g_{r_\ell^{\text{real}}}(r_\ell^{\text{real}}) \cdot \mathbf{G} = \mathbf{A}\mathbf{R}_{C_\ell} - \mathbf{G}$$

It follows that for all $x \in \{0, 1\}^\ell$, the first $(\ell - 1)$ matrices $\{\mathbf{B}_{C_i}\}_{i \in [\ell-1]}$ do not have a $\mathbf{G}$ -trapdoor since we use k^{fake} , but the matrix $\mathbf{B}_{C_\ell}$ will always have one because we use k^{real} . Consequently, the simulator uses $\mathbf{R}_{C_\ell}$ with $\mathbf{G}$ to sample a low-norm $\mathbf{R}_x$ such that,

$$[\mathbf{A} \parallel \mathbf{B}_{C_1} \parallel \cdots \parallel \mathbf{B}_{C_\ell}] \cdot \mathbf{R}_x = \mathbf{u}$$

while maintaining indistinguishability with respect to $\mathbf{R}_x$, when $\mathbf{R}_x$ is sampled in the real game with $\mathbf{T}_\mathbf{A}$, under properties of the $\mathbf{G}$ -trapdoor [MP12] for an appropriate choice of parameters.

To encrypt a random $\text{msg} \xleftarrow{\$} \{0, 1\}$ to a challenge string $y^* \in \{0, 1\}^\ell$, the simulator "samples" k^{enc} by fixing $k^{\text{enc}} = k^{\text{real}}$ and computes $r_i^{\text{enc}} = f(k^{\text{real}}, y^*(1, i))$ for $i \in [\ell]$. Then, for all i it computes,

$$\mathbf{B}_{f_{y^*, i}} + r_i^{\text{enc}} \cdot \mathbf{G} = \mathbf{A}\mathbf{R}_{f_{y^*, i}}$$

and also

$$\mathbf{c}_i^* = \mathbf{s}^\top \mathbf{A}\mathbf{R}_{f_{y^*, i}} + \mathbf{e}_i^\top \quad \text{for } i \in [\ell]$$

and outputs the ciphertext:

$$\text{ct}_{y^*} = (y^*, \{r_i^{\text{enc}}\}_{i \in [\ell]}, \mathbf{s}^\top \mathbf{A} + \mathbf{e}_a^\top, \{\mathbf{c}_i^*\}_{i \in [\ell]}, \mathbf{s}^\top \mathbf{u} + e_{\text{msg}} + \text{msg} \lceil q/2 \rceil).$$

We have to make sure the noise distribution in ct_{y^*} is the right one, but we defer to Section 4 about this. At this point the challenge ciphertext ct_{y^*} does not contain any " $\mathbf{G}$ -trapdoor", and we can use LWE to argue that msg remains hidden. As we argued at the start of this overview, during simulation the PRF values of the challenge ciphertext ct_{y^*} remain pseudorandom even given the PRF values of all requested secret keys sk_x , since these PRF values do not intersect. Thus, a PPT adversary cannot distinguish the real game from the simulated one with probability more than $1/2 + \text{negl}(\lambda, \ell_{\text{str}})$.

Extending functionality. The work of [CHK07] shows how to generically transform any string space with an arbitrary alphabet to $\{0, 1\}^*$ using a CRHF. Moreover, lattice-based PRF constructions [BPR12, BLMR13, BP14, Kim20] use public parameters that can be preprocessed outside the ABE evaluation, to improve the upper bound on the evaluation depth required to set the ABE parameters. Finally, our PRE scheme can be made *weakly-hiding* using obfuscation under LWE [GKW17, WZ17, CVW18, GKVW20]. That is, a ciphertext ct_y additionally hides y when given any secret key sk_x that is not authorized to decrypt ct_y .

1.3 Open Problems

The main open problem from our work is whether our scheme can be modified to support secret key extension in a secure way, as required by HIBE. In HIBE, a secret key sk_x for an identity $x \in \{0, 1\}^\ell$ can be extended to a key $\text{sk}_{x||x'}$ for $x' \in \{0, 1\}^{\ell'}$ without requiring the help of the authority. This allows to create a key hierarchy in a tree like manner. Our scheme can be slightly modified so that the keys can be extended using the basis delegation algorithms from [CHKP10, ABB10a, ABB10b, MP12]. To extend a key bit-by-bit, we would sample a fresh PRF seed to compute the circuits $C_{\ell+1}, \dots, C_{\ell+\ell'}$, then compute the matrices $\mathbf{B}_{C_{\ell+1}}, \dots, \mathbf{B}_{C_{\ell+\ell'}}$, and extend the key sk_x to a key $\text{sk}_{x||x'}$ for $\mathbf{B}_{C_1}, \dots, \mathbf{B}_{C_{\ell+\ell'}}$. In our case, to get meaningful HIBE security we have to consider the

Shi and Waters [SW08] security model, which differs only in the key request phase, as follows: the adversary makes three types of queries: secret key and extension queries for which it receives a pointer to the requested secret key that gets sampled (and stored) but not its value, and reveal queries that return the actual (stored) secret key value this time. Unfortunately, the key extension we sketched above does not satisfy this model, but that was our motivation behind issuing secret keys that carry their lineage in a bit-by-bit fashion.

Another interesting direction, is whether there are more applications of pre-computing on the public parameters before encryption, possibly using a key extension mechanism in some context other than HIBE.

2 Preliminaries

Notation. Let PPT denote probabilistic polynomial time. We use the notation $[n] \stackrel{\text{def}}{=} \{1, \dots, n\}$. All logarithms $\log(\cdot)$ are taken to the base 2. For a distribution or a random variable $\mathcal{D}$, we let $x \leftarrow \mathcal{D}$ denote the process of sampling x according to $\mathcal{D}$. Similarly for a randomized algorithm Alg we let $y \leftarrow \text{Alg}(x)$ denote the process of running $\text{Alg}(x)$ with fresh randomness and taking y as the output. For a finite set S we let $x \stackrel{\$}{\leftarrow} S$ denote sampling x uniformly at random from S . For a finite string $x \in \{0, 1\}^*$ we denote its length with $|x|$.

Unless stated otherwise, we use λ to denote the security parameter. We say that a function $g(\lambda)$ is negligible in λ , denoted by $\text{negl}(\lambda)$, if $g(\lambda) = o(1/\lambda^c)$ for all $c \in \mathbb{N}$. For random variables X, Y with support $\mathcal{X}, \mathcal{Y}$ respectively, we define the *statistical distance*

$$\text{SD}(X, Y) \stackrel{\text{def}}{=} \frac{1}{2} \sum_{u \in \mathcal{X} \cup \mathcal{Y}} |\Pr[X = u] - \Pr[Y = u]|.$$

We say that two ensembles of random variables $X = \{X_\lambda\}, Y = \{Y_\lambda\}$ are *statistically indistinguishable* and denote this with $X \stackrel{s}{\approx} Y$, if $\text{SD}(X_\lambda, Y_\lambda) = \text{negl}(\lambda)$. For a distribution ensemble $D = \{D_\lambda\}$ we let D_λ^m denote a vector of m samples, that are sampled independently from D_λ .

For any integer $q \geq 2$, we let $\mathbb{Z}_q$ denote the ring of integers modulo q . We represent elements of $\mathbb{Z}_q$ as integers in the range $(-q/2, q/2]$ and define the absolute value $|x|$ of $x \in \mathbb{Z}_q$ by taking its representative in this range.

Linear algebra. For a vector $\mathbf{v}$ let $\|\mathbf{v}\|$ and $\|\mathbf{v}\|_\infty$ denote its ℓ_2 and ℓ_∞ norm, respectively. For a matrix $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ where $\{\mathbf{x}_i\}_{i=1, \dots, n}$ is its ordered set of column vectors, denote $\|\mathbf{X}\| = \max_{1 \leq i \leq n} \|\mathbf{x}_i\|$, and $\|\mathbf{X}\|_\infty = \max_{1 \leq i \leq n} \|\mathbf{x}_i\|_\infty$. Let $\mathbf{X} \|\mathbf{X}'$ denote the ordered concatenation of the ordered column set of $\mathbf{X}$ and $\mathbf{X}'$. We write $\mathbf{F} = \{\mathbf{X}_1, \dots, \mathbf{X}_n\}$ to denote $\mathbf{F} = \mathbf{X}_1 \|\dots\| \mathbf{X}_n$. For an ordered set $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ of n linearly independent vectors in $\mathbb{R}^m$, denote as $\tilde{\mathbf{X}} = \{\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_n\}$ its *Gram-Schmidt orthogonalization*. For any matrix $\mathbf{X} \in \mathbb{Z}^{m \times m}$ we denote with $s_1(\mathbf{X})$ its largest singular value. It holds that $\|\tilde{\mathbf{X}}\| \leq \|\mathbf{X}\| \leq s_1(\mathbf{X}) \leq m \|\mathbf{X}\|_\infty$.

We require the following application of the Leftover Hash Lemma (LHL).

Lemma 2.1 (LHL [DORS08, ABB10a]). Let q be a prime and $m > (n+1) \log q + \omega(\log n)$. Let matrix $\mathbf{R} \stackrel{\$}{\leftarrow} \{-1, 1\}^{m \times k}$ where $k = k(n)$ is polynomial in n . Let matrix $\mathbf{A} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^{n \times m}$ and matrix $\mathbf{B} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^{n \times k}$. Then, the distribution $(\mathbf{A}, \mathbf{AR})$ is statistically close to the distribution $(\mathbf{A}, \mathbf{B})$.

2.1 Lattice Preliminaries

Lattices and cosets. For positive integers n, q and arbitrary $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, define the full-rank m -dimensional q -ary lattice, $\Lambda^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{Ax} = \mathbf{0} \bmod q\}$. For any $\mathbf{u} \in \mathbb{Z}_q^n$ define the

coset $\Lambda_{\mathbf{u}}^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{u} \bmod q\}$.

Discrete Gaussian distribution. The m -dimensional Gaussian distribution centered at zero with positive real parameter σ is defined by its probability density function: $\rho_\sigma(\mathbf{x}) = \exp(-\pi\|\mathbf{x}\|^2/\sigma^2)$. For a lattice $\Lambda \subset \mathbb{R}^m$ the discrete Gaussian distribution over Λ with parameter σ , is defined for all $\mathbf{x} \in \Lambda$ by $\mathcal{D}_{\Lambda,\sigma} = \rho_\sigma(\mathbf{x})/\sum_{\mathbf{x} \in \Lambda} \rho_\sigma(\mathbf{x})$, and takes the zero value for all other $\mathbf{x}$. We require the following lemma.

Lemma 2.2 ([MR07, Reg09, GPV08]). For the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^m,\sigma}$ over $\mathbb{Z}^m$ with parameter σ , it holds

$$\Pr[\|\mathbf{v}\| > \sigma\sqrt{m} : \mathbf{v} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}] = \text{negl}(m).$$

Bounded distributions and swallowing. We review the following from [BV16], that for our purposes apply to discrete Gaussian distributions over the integers.

For $\beta = \beta(\lambda)$ we say that a distribution ensemble $\chi = \{\chi_\lambda\}$ over $\mathbb{Z}$ is β -bounded if $\Pr[|x| > \beta : x \leftarrow \chi_\lambda] = \text{negl}(\lambda)$. For two bounded distribution ensembles $\chi = \{\chi_\lambda\}, \chi' = \{\chi'_\lambda\}$ over $\mathbb{Z}$, we say that χ' is *swallowing* for χ if $\chi' \stackrel{s}{\approx} \chi' + \chi$.

Corollary 2.3 ([BV16]). For a $\beta(\lambda)$ -bounded distribution ensemble $\chi = \{\chi_\lambda\}$ over $\mathbb{Z}$, and $\beta'(\lambda) = \beta(\lambda) \cdot \lambda^{\omega(1)}$, there exists an efficiently sampleable and $\beta'(\lambda)$ -bounded ensemble $\chi' = \{\chi'_\lambda\}$ over $\mathbb{Z}$ that is swallowing for χ .

Short Integer Solution. For positive integers n, m, q and positive real β , the short integer solution (SIS) problem for a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ asks to find a *non-zero* vector $\mathbf{x} \in \mathbb{Z}_q^m$ such that $\mathbf{A}\mathbf{x} = \mathbf{0} \bmod q$ and $\|\mathbf{x}\| \leq \beta$.

The works of [Ajt96, MR07, GPV08, MP13] show that for any $m = \text{poly}(n)$, $\beta > 0$ and sufficiently large $q \geq \beta \cdot \text{poly}(n)$ the (n, m, q, β) -SIS problem is as hard as approximating certain worst-case lattice problems such as GapSVP and SIVP on n -dimensional lattices to within a $\beta \cdot \text{poly}(n)$ factor.

Learning with Errors. The learning with errors (LWE) assumption was introduced by Regev in [Reg05] and states the following. Let n, q be positive integers and χ be a β -bounded distribution on $\mathbb{Z}_q$, all parameterized by the security parameter λ . The LWE assumption says that for all polynomial m the following distributions are computationally indistinguishable

$$(\mathbf{A}, \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top) \stackrel{c}{\approx} (\mathbf{A}, \mathbf{u}^\top) : \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}, \mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi^m, \mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^m.$$

The works of [Reg05, Pei09, ACPS09, MM11, MP12, BLP⁺13, PRS17] show that the (n, m, q, χ) -LWE assumption is as hard as (quantumly) approximating certain worst-case lattice problems such as GapSVP and SIVP on n -dimensional lattices to within a $\tilde{O}(n \cdot q/\beta)$ factor.

Lattice trapdoors. We require the following lemma which states that we can sample lattices with "trapdoors" that are statistically indistinguishable from random lattices, and the "trapdoor" allows to efficiently solve the SIS problem.

Lemma 2.4 (Lattice trapdoors [Ajt99, GPV08, AP09, MP12, LW15]). Let λ be the security parameter with lattice parameters n, m, q and norm bound β all parameterized by λ where $n = \lambda^{\Omega(1)}$, $m = \Omega(n \log q)$ and $\sigma \geq \|\tilde{\mathbf{T}}_{\mathbf{A}}\| \cdot \omega(\sqrt{\log m})$. Then, there exists a tuple of efficient algorithms (TrapGen, SamPre, Sam) with the following syntax:

- $\text{TrapGen}(1^n, 1^m, q) \rightarrow (\mathbf{A}, \mathbf{T}_\mathbf{A})$. The TrapGen algorithm takes as input the matrix dimensions n, m and modulus q , and outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and trapdoor $\mathbf{T}_\mathbf{A} \in \mathbb{Z}^{m \times m}$.
- $\text{Sam}(1^m, q) \rightarrow \mathbf{C}$. The Sam algorithm takes as input matrix dimension m and modulus q , and outputs a matrix $\mathbf{C} \in \mathbb{Z}^{m \times m}$.
- $\text{SamPre}(\mathbf{A}, \mathbf{U}, \mathbf{T}_\mathbf{A}, \sigma) \rightarrow \mathbf{C}$. The SamPre algorithm takes as input matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{U} \in \mathbb{Z}_q^{n \times m}$, trapdoor $\mathbf{T}_\mathbf{A}$ and parameter σ , and outputs a matrix $\mathbf{C} \in \mathbb{Z}^{m \times m}$ such that $\mathbf{AC} = \mathbf{U}$.

Furthermore, the following properties are satisfied:

1. For any $(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \text{TrapGen}(1^n, 1^m, q)$, $\mathbf{U} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{C} \leftarrow \text{SamPre}(\mathbf{A}, \mathbf{U}, \mathbf{T}_\mathbf{A})$ we have $\mathbf{AC} = \mathbf{U}$ and $\|\mathbf{C}\| \leq \sqrt{m} \cdot \sigma$ (with probability 1).
2. We have the *statistical indistinguishability* requirement $\mathbf{A} \stackrel{s}{\approx} \mathbf{A}'$ where $(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \text{TrapGen}(1^n, 1^m, q)$ and $\mathbf{A}' \xleftarrow{s} \mathbb{Z}_q^{n \times m}$.
3. We have the *statistical indistinguishability* requirement $(\mathbf{A}, \mathbf{T}_\mathbf{A}, \mathbf{C}) \stackrel{s}{\approx} (\mathbf{A}, \mathbf{T}_\mathbf{A}, \mathbf{C}')$ where $(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \text{TrapGen}(1^n, 1^m, q)$, $\mathbf{C} \leftarrow \text{Sam}(1^m, q)$, $\mathbf{U} \xleftarrow{s} \mathbb{Z}_q^{n \times m}$ and $\mathbf{C}' \leftarrow \text{SamPre}(\mathbf{A}, \mathbf{U}, \mathbf{T}_\mathbf{A})$.

Gadget matrix. In this work we use the gadget lattice trapdoors formalized in [MP12]. Given integers $n \geq 1, q \geq 2$, the gadget matrix is defined as $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}$ where $\mathbf{g} = (1, 2, 4, \dots, 2^{n \times n \cdot \lceil \log q \rceil})$. The bit-decomposition function $\mathbf{G}^{-1} : \mathbb{Z}_q^{n \times m} \rightarrow \{0, 1\}^{n \cdot \lceil \log q \rceil \times m}$ takes as input a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and expands its every entry $a_{i,j} \in \mathbb{Z}_q$ to its binary decomposition in a column-wise manner. For simplicity we will assume that $\mathbf{G}$ has width m (otherwise we pad with zeros), and thus $\mathbf{G}^{-1}$ has range $\{0, 1\}^{m \times m}$. For all $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ it holds that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$.

Gadget trapdoors. A $\mathbf{G}$ -trapdoor (or simply trapdoor) of a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times 2m}$ is a full-rank (over the integers) matrix $\mathbf{T}_\mathbf{A} = [\mathbf{R} \ \mathbf{I}_m]^T$ where $\mathbf{R} \in \mathbb{Z}^{m \times m}$ is a low-norm matrix, such that $\mathbf{AT}_\mathbf{A} = \mathbf{G}$. Specifically, $\mathbf{A} = [\mathbf{A}' | \mathbf{A}'\mathbf{R} + \mathbf{G}]$ for $\mathbf{A}' \xleftarrow{s} \mathbb{Z}_q^{n \times m}$.

We require the following lemma that is used to sample an extended lattice basis.

Lemma 2.5 (Lattice basis sampling (adapted) [GPV08, AP09, CHKP10, ABB10a, ABB10b, MP12]). Let λ be the security parameter with lattice parameters n, m, q and norm bound σ all parameterized by λ where $n = \lambda^{\Omega(1)}$, $m = \Omega(n \log q)$, and $d \geq 1$. Then, there exists a tuple of efficient algorithms ($\text{SampleBasisLeft}, \text{SampleBasisRight}$) with the following properties:

- $\text{SampleBasisLeft}(\mathbf{A}, \mathbf{B}, \mathbf{T}_\mathbf{A}, \sigma) \rightarrow \mathbf{T}_{\mathbf{A}||\mathbf{B}}$. The SampleBasisLeft algorithm takes as input matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{B} = \{\mathbf{B}_i\}_{i \in [d]} \in \mathbb{Z}_q^{n \times dm}$ with $\mathbf{B}_i \in \mathbb{Z}_q^{n \times m}$, a trapdoor $\mathbf{T}_\mathbf{A} \in \mathbb{Z}^{m \times m}$ and a norm bound σ , and outputs a basis $\mathbf{T}_{\mathbf{A}||\mathbf{B}} \in \mathbb{Z}^{(d+1)m \times m}$ for $\Lambda^\perp(\mathbf{A}||\mathbf{B})$.

For any positive real $\sigma \geq \|\tilde{\mathbf{T}}_\mathbf{A}\| \cdot \omega(\sqrt{\log(dm)})$ it holds that $\|\tilde{\mathbf{T}}_\mathbf{A}\| = \|\tilde{\mathbf{T}}_{\mathbf{A}||\mathbf{B}}\|$.

For any two $\mathbf{T}_\mathbf{A}, \mathbf{T}'_\mathbf{A}$ and $\sigma \geq \max(\|\tilde{\mathbf{T}}_\mathbf{A}\|, \|\tilde{\mathbf{T}}'_\mathbf{A}\|) \cdot \omega(\sqrt{\log(dm)})$, then for the *same* lattice, the output distributions of SampleBasisLeft in this case are within $\text{negl}(\lambda)$ statistical distance.

- $\text{SampleBasisRight}(\mathbf{A}, \mathbf{B}, \mathbf{R}, \mathbf{G}, \sigma) \rightarrow \mathbf{T}_{\mathbf{A}||\mathbf{B}}$ takes as input matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{B} = \{\mathbf{B}_i\}_{i \in [d]} \in \mathbb{Z}_q^{n \times dm}$ with $\mathbf{B}_i \in \mathbb{Z}_q^{n \times m}$, a matrix $\mathbf{R} \in \mathbb{Z}^{m \times m}$, the gadget matrix $\mathbf{G}$ with $\mathbf{B}_d = \mathbf{AR} - \mathbf{G}$, and a norm bound σ and outputs a basis $\mathbf{T}_{\mathbf{A}||\mathbf{B}} \in \mathbb{Z}^{(d+1)m \times m}$ for $\Lambda^\perp(\mathbf{A}||\mathbf{B})$.

Note that $\mathbf{T}_\mathbf{F} = \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix}$ can be made into a $\mathbf{G}$ -trapdoor for $\Lambda^\perp(\mathbf{F})$ with $\mathbf{F} = \mathbf{A} \parallel \mathbf{B}_d$ and then into a basis for $\mathbf{A} \parallel \mathbf{B}$ in a similar manner to above. Thus, for any positive real $\sigma \geq \|\tilde{\mathbf{T}}_\mathbf{F}\| \cdot \omega(\sqrt{\log(dm)})$ it holds that $\|\tilde{\mathbf{T}}_\mathbf{F}\| = \|\tilde{\mathbf{T}}_{\mathbf{A} \parallel \mathbf{B}}\|$.

For any two $\mathbf{T}_\mathbf{F}, \mathbf{T}'_\mathbf{F}$ for the *same* lattice, and $\sigma \geq \max(\|\tilde{\mathbf{T}}_\mathbf{F}\|, \|\tilde{\mathbf{T}}'_\mathbf{F}\|) \cdot \omega(\sqrt{\log(dm)})$, the output distributions of `SampleBasisRight` in this case are within $\text{negl}(\lambda)$ statistical distance.

Finally, there is an efficient algorithm (`SamPre`) that takes as input $\mathbf{F} = \mathbf{A} \parallel \mathbf{B} \in \mathbb{Z}_q^{n \times (d+1)m}$, a basis $\mathbf{T}_\mathbf{F} \in \mathbb{Z}^{(d+1)m \times m}$ for $\Lambda^\perp(\mathbf{F})$, any $\mathbf{u} \in \mathbb{Z}_q^n$ and parameter $\sigma \geq \|\tilde{\mathbf{T}}_\mathbf{F}\| \cdot \omega(\sqrt{\log(dm)})$, and samples $\mathbf{c} \in \mathbb{Z}^{(d+1)m}$ with $\|\mathbf{c}\| \leq \sigma \sqrt{(d+1)m}$ and within $\text{negl}(\lambda)$ statistical distance from $\mathcal{D}_{\Lambda^\perp_\mathbf{F}(\mathbf{F}), \sigma}$, such that $\mathbf{F} \cdot \mathbf{c} = \mathbf{u} \bmod q$.

Proposition 2.6. [Lattice basis sampling indistinguishability (adapted) [CHKP10, ABB10a, ABB10b]] The output distributions of algorithms `SampleBasisLeft`, `SampleBasisRight` of Lemma 2.5 when given as input the *same* lattice, are within negligible statistical distance when $\sigma > \max(s_1(\mathbf{T}_\mathbf{A}), s_1(\mathbf{T}_\mathbf{F})) \cdot \omega(\sqrt{\log(dm)})$.

2.2 Homomorphic Evaluation

We require the following lemma about homomorphic evaluation on ciphertexts and public keys that have a specific structure.

Lemma 2.7 ([GSW13, BGG⁺14]). Let λ be the security parameter with lattice parameters n, m, q and norm bound β all parameterized by λ where $n = \lambda^{\Omega(1)}$ and $m = \Omega(n \log q)$. Let $f : \{0, 1\}^\ell \rightarrow \{0, 1\}$ be any circuit of depth at most d .

Then, there exists a tuple of efficient *deterministic* algorithms (`EvalCT`, `EvalPK`, `EvalSimPK`) with the following properties:

- `EvalCT` ($f, \{\mathbf{B}_i, x_i, \mathbf{c}_i\}_{i \in [\ell]} \rightarrow \mathbf{c}_f$. The `EvalCT` algorithm takes as input the description of a circuit f , matrices $\mathbf{B}_i \in \mathbb{Z}_q^{n \times m}$, the input bits $x_i \in \{0, 1\}$ for f , and vectors $\mathbf{c}_i$ such that

$$\mathbf{c}_i = \mathbf{s}^\top (\mathbf{B}_i + x_i \mathbf{G}) + \mathbf{e}_i^\top \quad \text{with} \quad \|\mathbf{e}_i\|_\infty \leq \beta.$$

It outputs a vector $\mathbf{c}_f = \mathbf{s}^\top (\mathbf{B}_f + f(x) \cdot \mathbf{G}) + \mathbf{e}_f^\top$, where the matrix $\mathbf{B}_f$ depends only on $\mathbf{B}_i$ and f , and $\|\mathbf{e}_f\|_\infty \leq m^{O(d)} \cdot \beta$.

- `EvalPK` ($f, \{\mathbf{B}_i\}_{i \in [\ell]} \rightarrow \mathbf{B}_f$. The `EvalPK` algorithm takes as input the description of a circuit f and matrices $\mathbf{B}_i \in \mathbb{Z}_q^{n \times m}$, and outputs a matrix $\mathbf{B}_f \in \mathbb{Z}_q^{n \times m}$.
- `EvalSimPK` ($f, \mathbf{A}, \{\mathbf{B}_i, \mathbf{R}_i, x_i\}_{i \in [\ell]} \rightarrow \mathbf{R}_f$. The `EvalSimPK` algorithm takes as input the description of a circuit f , matrices $\mathbf{A}, \mathbf{B}_i \in \mathbb{Z}_q^{n \times m}$, matrices $\mathbf{R}_i \in \{-1, 1\}^{m \times m}$, and the input bits $x_i \in \{0, 1\}$ for f such that

$$\mathbf{B}_i = \mathbf{A} \mathbf{R}_i - x_i \mathbf{G}$$

It outputs a matrix $\mathbf{R}_f$ such that $\mathbf{B}_f = \mathbf{A} \mathbf{R}_f - f(x) \cdot \mathbf{G}$, and $\|\mathbf{R}_f\|_\infty \leq m^{O(d)}$, where $\mathbf{B}_f = \text{EvalPK}(f, \{\mathbf{B}_i\}_{i \in [\ell]})$.

For any $x \in \{0, 1\}^\ell$ and $f : \{0, 1\}^\ell \rightarrow \{0, 1\}$, $g : \{0, 1\}^{\ell_h} \rightarrow \{0, 1\}$ and $h : \{0, 1\}^\ell \rightarrow \{0, 1\}^{\ell_h}$, where $f = g \circ h$ and f is described by a circuit that is gate-by-gate identical to $g \circ h$, then the above algorithms respect composition. We denote with h_j the circuit that computes the j -th output bit of h for $j \in [\ell_h]$. That is,

- $\text{EvalCT}(f, \{\mathbf{B}_i, x_i, \mathbf{c}_i\}_{i \in [\ell]}) = \text{EvalCT}(g, \{\mathbf{B}_{h_j}, h_j(x), \text{EvalCT}(h_j, \{\mathbf{B}_i, x_i, \mathbf{c}_i\}_{i \in [\ell]})\}_{j \in [\ell_h]})$
with $\mathbf{B}_{h_j} = \text{EvalPK}(h_j, \{\mathbf{B}_i\}_{i \in [\ell]})$ or $\mathbf{B}_{h_j} = \text{EvalSimPK}(h_j, \mathbf{A}, \{\mathbf{B}_i, \mathbf{R}_i, x_i\}_{i \in [\ell]})$,
depending on the context.
- $\text{EvalPK}(f, \{\mathbf{B}_i\}_{i \in [\ell]}) = \text{EvalPK}(g, \{\text{EvalPK}(h_j, \{\mathbf{B}_i\}_{i \in [\ell]})\}_{j \in [\ell_h]})$
- $\text{EvalSimPK}(f, \mathbf{A}, \{\mathbf{B}_i, \mathbf{R}_i, x_i\}_{i \in [\ell]}) = \text{EvalSimPK}(g, \mathbf{A}, \{\mathbf{B}_{h_j}, \mathbf{R}_{h_j}, h_j(x)\}_{j \in [\ell_h]})$
with $\mathbf{R}_{h_j} = \text{EvalSimPK}(h_j, \mathbf{A}, \{\mathbf{B}_i, \mathbf{R}_i, x_i\}_{i \in [\ell]})$

Remark 2.8. If we only use NAND gates then we *implicitly* assume in the public parameters a matrix $\mathbf{B}_0$ (real game) or $\mathbf{B}_0 = \mathbf{A}\mathbf{R}_0 - \mathbf{G}$ (simulated game) that encodes then (constant) bit 1 as $\mathbf{B}_0 + \mathbf{G}$ and is given as input to the above algorithms, as required for evaluation correctness (see [KNYY20, Remark 3.2]). If we only use XOR and AND gates this is not required (see [BGG⁺14], [Pei16, Section 6.2.2]).

2.3 Pseudorandom Functions

We review the notion of a pseudorandom function family [GGM84] adapted for our PRE scheme. Lattice-based PRF constructions [BPR12, BLMR13, BP14, Kim20] require a set of *random* public parameters $\mathbf{pp}$ that are known to all parties, including the adversary. These are generated by a trusted authority. Moreover, for every positive integer $\ell_{\text{str}} = \ell_{\text{str}}(\lambda)$ we assume an efficient procedure $\text{Encode}_{\ell_{\text{str}}} : \{0, 1\}^{\ell_{\text{str}}} \times \{0, 1\}^{\lceil \log(\ell_{\text{str}}) \rceil} \rightarrow \mathcal{X}_\lambda$ that is part of the PRF.Eval algorithm, and maps *prefix strings* to the input domain of the PRF using standard techniques. We can safely assume ℓ_{str} to be upper bounded by a polynomial in λ and thus $\text{Encode}_{\ell_{\text{str}}}$ is efficient with respect to λ .

Definition 2.9. A pseudorandom function (PRF) family $\Pi^{\text{PRF}, \text{PRE}} = \{\Pi_{\lambda, \ell_{\text{str}}}^{\text{PRF}, \text{PRE}}\}$ with parameter space $\mathcal{PP} = \{\mathcal{PP}_\lambda\}$, key space $\mathcal{K} = \{\mathcal{K}_\lambda\}$, domain space $\mathcal{X} = \{\mathcal{X}_\lambda\}$ and range space $\mathcal{Y} = \{\mathcal{Y}_\lambda\}$ consists of a tuple of efficient algorithms $\Pi_{\lambda, \ell_{\text{str}}}^{\text{PRF}, \text{PRE}} = (\text{PRF.Param}, \text{PRF.Keygen}, \text{PRF.Eval})$ with the following syntax and security:

- $\text{PRF.Param}(1^\lambda, 1^{\ell_{\text{str}}}) \rightarrow \mathbf{pp}$. The PRF.Param algorithm takes as input the security parameter λ and outputs the public parameters $\mathbf{pp} \in \mathcal{PP}_\lambda$.
- $\text{PRF.Keygen}(1^\lambda) \rightarrow k$. The PRF.Keygen algorithm takes as input the security parameter λ and outputs a key $k \in \mathcal{K}_\lambda$.
- $\text{PRF.Eval}(\mathbf{pp}, k, x) \rightarrow y$. The PRF.Eval algorithm takes as input the public parameters $\mathbf{pp}$, a key k and a string x , and outputs a value $y \in \mathcal{Y}_\lambda$.

The algorithm PRF.Eval is a proxy for running $\text{PRF.Eval}'(\mathbf{pp}, k, \text{Encode}_{\ell_{\text{str}}}(x, |x|)) \rightarrow y$, where $|x| \in \{0, 1\}^{\lceil \log(\ell_{\text{str}}) \rceil}$, i.e. $|x|$ is represented in binary with zero padding if necessary. The $\text{PRF.Eval}'$ is the actual PRF evaluation algorithm that expects an input of length equal to the output length of $\text{Encode}_{\ell_{\text{str}}}$.

Security. For every PPT adversary $\mathcal{A}$, we require that the following pseudorandomness property is satisfied:

$$\left| \Pr \left[\mathcal{A}^{\text{PRF.Eval}(\mathbf{pp}, k, \cdot)}(1^\lambda, \mathbf{pp}) = 1 \mid \begin{array}{l} \mathbf{pp} \leftarrow \text{PRF.Param}(1^\lambda) \\ k \leftarrow \text{PRF.Keygen}(1^\lambda) \end{array} \right] - \Pr \left[\mathcal{A}^{\mathcal{R}_\lambda(\cdot)}(1^\lambda, \mathbf{pp}) = 1 \right] \right| = \text{negl}(\lambda)$$

where $\mathcal{R}_\lambda : \mathcal{X}_\lambda \rightarrow \mathcal{Y}_\lambda$ is a (truly) random function.

For security, we also assume that every PPT (admissible) adversary $\mathcal{A}$ outputs valid tuples $(x, i) \in (\{0, 1\}^{\ell_{\text{str}}} \times \{0, 1\}^{\lceil \log(\ell_{\text{str}}) \rceil})$ (i.e. such that $i = |x|$) to which the procedure $\text{Encode}_{\ell_{\text{str}}}$ is applied before given as input to $\text{PRF.Eval}'(\text{pp}, k, \cdot)$ or $\mathcal{R}_\lambda(\cdot)$. This allows us to incorporate the only security property we require from $\text{Encode}_{\ell_{\text{str}}}$, namely that for every PPT adversary $\mathcal{A}(1^\lambda, 1^{\ell_{\text{str}}})$ it remains $\text{negl}(\lambda, \ell_{\text{str}})$ to output two different tuples $(x, i), (x', i')$ such that $\text{Encode}_{\ell_{\text{str}}}(x, i) = \text{Encode}_{\ell_{\text{str}}}(x', i')$, e.g. assuming collision-resistance and an error-correcting code. Thus, the probability for the pseudorandomness property is taken over all randomness involved in the experiment.

3 PRE Definitions

In this section, we define the syntax for a PRE scheme, and state the correctness and security properties.

Definition 3.1 (PRE). A PRE scheme that supports strings of a-priori bounded length $\ell_{\text{str}} = \ell_{\text{str}}(\lambda)$ consists of the following four PPT algorithms:

- **Setup** $(1^\lambda, 1^{\ell_{\text{str}}}) \rightarrow (\text{pp}, \text{msk})$. The Setup algorithm takes as input the security parameter λ and a maximum string length ℓ_{str} , and outputs the public parameters pp and the master secret key msk .
- **Keygen** $(\text{pp}, \text{msk}, x) \rightarrow \text{sk}_x$. The key generation algorithm takes as input the public parameters pp , the master secret key msk and a string x , and outputs a secret key sk_x .
- **Enc** $(\text{pp}, y, m) \rightarrow \text{ct}_y$. The encryption algorithm takes as input the public parameters pp , a string y and a message m , and outputs a ciphertext ct_y .
- **Dec** $(\text{pp}, \text{ct}_y, \text{sk}_x) \rightarrow m$ or $\perp$. The decryption algorithm takes as input the public parameters pp , a ciphertext ct_y and a secret key sk_x , and outputs a message m or the special symbol $\perp$ that denotes decryption failure.

We require the following two properties on correctness and security.

Correctness. For all positive integers $\lambda, \ell_{\text{str}}$, and $x, y \in \{0, 1\}^{\ell_{\text{str}}}$ where x is a prefix of y and $m \in \{0, 1\}$, we require:

$$\Pr \left[\text{Dec}(\text{pp}, \text{ct}_y, \text{sk}_x) = m \mid \begin{array}{l} (\text{pp}, \text{msk}) \leftarrow \text{Setup}(1^\lambda, 1^{\ell_{\text{str}}}) \\ \text{ct}_y \leftarrow \text{Enc}(\text{pp}, y, m) \\ \text{sk}_x \leftarrow \text{Keygen}(\text{pp}, \text{msk}, x) \end{array} \right] = 1 - \text{negl}(\lambda, \ell_{\text{str}}).$$

Security. We define the following adaptive security game between a PPT challenger and a PPT adversary $\mathcal{A}$.

Setup. The challenger runs the **Setup** $(1^\lambda, 1^{\ell_{\text{str}}})$ algorithm to generate the public parameters pp and the master secret key msk , and gives pp to the adversary $\mathcal{A}$, including ℓ_{str} .

Phase 1. The adversary $\mathcal{A}$ adaptively submits secret key requests for any string x . For each request, the challenger responds with a secret key sk_x by running **Keygen** $(\text{pp}, \text{msk}, x)$.

Challenge. The adversary $\mathcal{A}$ submits a challenge string y^* under the restriction that none of the requested keys in **Phase 1** is a prefix of y^* . The challenger samples $m \xleftarrow{\$} \{0, 1\}$ and encrypts m to y^* by running **Enc** (pp, y^*, m) , and sends ct_{y^*} to the adversary $\mathcal{A}$.

Phase 2. The adversary $\mathcal{A}$ adaptively submits secret keys requests for any string x under the restriction that x is not a prefix of y^* . The challenger responds with a secret key sk_x by running $\text{Keygen}(\text{pp}, \text{msk}, x)$.

Guess. The adversary outputs a bit $m' \in \{0, 1\}$.

For a PPT adversary $\mathcal{A}$ its advantage $\text{Adv}_{\mathcal{A}}^{\text{PRE}}(1^\lambda, 1^{\ell_{\text{str}}})$ in the above security game is defined as $|\Pr[m' = m] - 1/2|$. A PRE scheme is adaptively secure if for every PPT adversary $\mathcal{A}$ it holds that $\text{Adv}_{\mathcal{A}}^{\text{PRE}}(1^\lambda, 1^{\ell_{\text{str}}}) = \text{negl}(\lambda, \ell_{\text{str}})$.

We can safely assume that ℓ_{str} is bounded by some polynomial in λ . We explicitly provide ℓ_{str} during **Setup** to instantiate the parameters for the $\text{Encode}_{\ell_{\text{str}}}$ procedure.

4 Adaptively Secure PRE

In this section, we describe our adaptively secure PRE scheme and prove the following theorem.

Theorem 4.1. Under the polynomial hardness of approximating SIVP and GapSVP on n -dimensional lattices to within a sub-exponential factor of 2^{n^ε} for some $\varepsilon \in (0, 1/2)$ where $n = n(\lambda)$, there exists an adaptively secure PRE scheme that satisfies Definition 3.1.

Proof. The proof follows from Lemma 4.2 and Lemma 4.3. □

4.1 PRE Construction

In this section, we describe our PRE construction in two steps. First, we define three circuits used in the construction, following closely [Tsa19] and then give the details of the construction.

4.1.1 Circuit Definitions

Fix $\lambda, \ell_{\text{str}}$ and string space $\{0, 1\}^{\ell_{\text{str}}}$. Fix a PRF family $\Pi^{\text{PRF}, \text{PRE}} = \{\Pi_{\lambda, \ell_{\text{str}}}^{\text{PRF}, \text{PRE}}\}$ with $\Pi_{\lambda, \ell_{\text{str}}}^{\text{PRF}, \text{PRE}} = (\text{PRF.Param}, \text{PRF.Keygen}, \text{PRF.Eval})$ as defined in Section 2.3. In this section, we rely on the binary representation of certain parameters and for that reason let values $\ell_k = \ell_k(\lambda)$, $\ell_{\text{in}} = \ell_{\text{in}}(\lambda)$, $\ell_{\text{out}} = \ell_{\text{out}}(\lambda)$ be such that it holds: $\mathbf{K}_\lambda = \{0, 1\}^{\ell_k}$, $\mathbf{X}_\lambda = \{0, 1\}^{\ell_{\text{in}}}$, $\mathbf{Y}_\lambda = \{0, 1\}^{\ell_{\text{out}}}$. Finally, let PP_λ be the parameter space. To simplify notation and without explicitly stating it, we assume that all hardcoded values and inputs to the circuits defined below are given in binary representation.

- The circuit $f_{x,i}^{\text{prf}} : \mathbf{K}_\lambda \rightarrow \mathbf{Y}_\lambda$ has *hardcoded* the $\Pi_{\lambda, \ell_{\text{str}}}^{\text{PRF}, \text{PRE}}$ parameters $\text{pp} \in \text{PP}_\lambda$ and the *prefix* $x(1, i)$ of a string $x \in \{0, 1\}^{\ell_{\text{str}}}$ for some $i \leq \ell_{\text{str}}$. It takes as input a $\Pi_{\lambda, \ell_{\text{str}}}^{\text{PRF}, \text{PRE}}$ key $k \in \mathbf{K}_\lambda$ and outputs the $\Pi_{\lambda, \ell_{\text{str}}}^{\text{PRF}, \text{PRE}}$ value:

$$f_{x,i}^{\text{prf}}(k) = \text{PRF.Eval}(\text{pp}, k, x(1, i))$$

- The circuit $f_r^{\text{eq}} : \mathbf{Y}_\lambda \rightarrow \{0, 1\}$ has *hardcoded* a value $r \in \mathbf{Y}_\lambda$. It takes as input a value $y \in \mathbf{Y}_\lambda$ and outputs:

$$f_r^{\text{eq}}(y) = \begin{cases} 1, & \text{if } y = r \\ 0, & \text{otherwise} \end{cases}$$

- The circuit $f_i^{\text{comp}} : K_\lambda \rightarrow \{0, 1\}$ is the *composition* of circuits $f_{x,i}^{\text{prf}}$ and f_r^{eq} . That is, $f_i^{\text{comp}} = f_r^{\text{eq}} \circ f_{x,i}^{\text{prf}}$. It has *hardcoded* the $\Pi_{\lambda, \ell_{\text{str}}}^{\text{PRF,PRE}}$ parameters $\text{pp} \in \text{PP}_\lambda$, the *prefix* $x(1, i)$ of a fixed string $x \in \{0, 1\}^{\ell_{\text{str}}}$ for some $i \leq \ell_{\text{str}}$ and a value $r \in Y_\lambda$. It takes as input a $\Pi_{\lambda, \ell_{\text{str}}}^{\text{PRF,PRE}}$ key $k \in K_\lambda$ and outputs:

$$f_{x,i,r}^{\text{comp}}(k) = f_r^{\text{eq}}\left(f_{x,i}^{\text{prf}}(k)\right)$$

Lattice-based PRF constructions in [BPR12, BLMR13, BP14, Kim20] can be instantiated with $K_\lambda = \{0, 1\}^{\ell_k}$, $X_\lambda = \{0, 1\}^{\ell_{\text{in}}}$, $Y_\lambda = \{0, 1\}^{\ell_{\text{out}}}$. For example in [BP14] (similarly for the rest of these works), for suitable positive integers n, m, p, q , a PRF key is a vector in $\mathbb{Z}_q^n$, and the PRF public parameters are two matrices in $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}$. Thus, we can use the binary decomposition of these quantities that the algorithm PRF.Eval will take into account (along with n, m, p, q). Similarly, the PRF output is a vector in $\mathbb{Z}_p^m$, and the algorithm PRF.Eval could output its binary decomposition instead.

4.1.2 PRE Construction

Fix a security parameter λ , $n = n(\lambda)$, $m = m(\lambda)$, $q = q(\lambda)$ and length bound $\ell_{\text{str}} = \ell_{\text{str}}(\lambda)$. Fix a PRF family $\Pi^{\text{PRF,PRE}} = \{\Pi_{\lambda, \ell_{\text{str}}}^{\text{PRF,PRE}}\}$ with $\Pi_{\lambda, \ell_{\text{str}}}^{\text{PRF,PRE}} = (\text{PRF.Param}, \text{PRF.Keygen}, \text{PRF.Eval})$ as defined in Section 2.3 with binary representation of some parameters specified in Section 4.1.1, and circuits $f_{x,i}^{\text{prf}}, f_r^{\text{eq}}, f_i^{\text{comp}}$ as defined in Section 4.1.1. Our PRE scheme consists of a tuple of four efficient algorithms ($\text{Setup}, \text{Keygen}, \text{Enc}, \text{Dec}$) with the properties listed below. Note that we explicitly provide the $\Pi_{\lambda, \ell_{\text{str}}}^{\text{PRF,PRE}}$ key bit-length ℓ_k (as defined in Section 4.1.1) as input for the Setup algorithm to sample accordingly the public parameters.

- $\text{Setup}(1^\lambda, 1^{\ell_{\text{str}}}, 1^{\ell_k})$. On input the security parameter λ , maximum string length ℓ_{str} and key length ℓ_k , do the following:

- sample $(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \text{TrapGen}(1^n, 1^m, q)$
- sample $\mathbf{B}_i \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ for $i \in [\ell_k]$ and $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$
- sample $\text{pp}_{\text{prf}} \leftarrow \text{PRF.Param}(1^\lambda, 1^{\ell_{\text{str}}})$ and $k^{\text{real}}, k^{\text{fake}} \leftarrow \text{PRF.Keygen}(1^\lambda)$

Output

$$\text{pp} = (\text{pp}_{\text{prf}}, \mathbf{u}, \mathbf{A}, \mathbf{B}_1, \dots, \mathbf{B}_{\ell_k}) \quad \text{msk} = (\mathbf{T}_\mathbf{A}, k^{\text{real}}, k^{\text{fake}})$$

Output parameters $n, m, q, \chi, \widehat{\chi}, \{\sigma_i\}_{i \in [\ell_{\text{str}}]}$ as defined in Section 4.2.

- $\text{Keygen}(\text{pp}, \text{msk}, x)$. On input the public parameters pp , the master secret key msk and string $x \in \{0, 1\}^\ell$ with $\ell \leq \ell_{\text{str}}$, do the following:

- set $r_i = \text{PRF.Eval}(\text{pp}_{\text{prf}}, k^{\text{fake}}, x(1, i))$ for $i \in [\ell - 1]$
- set $r_\ell = \text{PRF.Eval}(\text{pp}_{\text{prf}}, k^{\text{real}}, x(1, \ell))$
- set $f_i^{\text{comp}} = f_{x,i}^{\text{prf}} \circ f_{r_i}^{\text{eq}}$ for $i \in [\ell]$
- set $\mathbf{B}_{f_i^{\text{comp}}} = \text{EvalPK}(f_i^{\text{comp}}, \{\mathbf{B}_\kappa\}_{\kappa \in [\ell_k]})$ for $i \in [d]$
- set $\mathbf{T}_x \leftarrow \text{SampleBasisLeft}(\mathbf{A}, \{\mathbf{B}_{f_i^{\text{comp}}}\}_{i \in [\ell]}, \mathbf{T}_\mathbf{A}, \sigma_\ell)$
- set $\mathbf{F}_x = \mathbf{A} \parallel \mathbf{B}_{f_1^{\text{comp}}} \parallel \dots \parallel \mathbf{B}_{f_\ell^{\text{comp}}}$

– sample $\mathbf{t}_{\mathbf{u},x}^{\text{comp}} \leftarrow \text{SamPre}(\mathbf{F}_x, \mathbf{T}_x, \mathbf{u}, \sigma_\ell)$

Output $\text{sk}_x = (x, \mathbf{t}_{\mathbf{u},x}^{\text{comp}}, \{r_i\}_{i \in [\ell]})$

- $\text{Enc}(\text{pp}, y, \text{msg})$. On input the public parameters pp , a string $y \in \{0, 1\}^\ell$ with $\ell \leq \ell_{\text{str}}$ and message $\text{msg} \in \{0, 1\}$, do the following:

- sample $k^{\text{enc}} \leftarrow \text{PRF.Keygen}(1^\lambda)$
- set $r_i^{\text{enc}} = \text{PRF.Eval}(\text{pp}_{\text{prf}}, k^{\text{enc}}, y(1, i))$ for $i \in [\ell]$
- set $C_{i,j}^{\text{prf}}$ to be the circuit that computes the j -th output bit of $f_{x,i}^{\text{prf}}$
- set $\mathbf{B}_{i,j}^{\text{prf}} = \text{EvalPK}(C_{i,j}^{\text{prf}}, \{\mathbf{B}_\kappa\}_{\kappa \in [\ell_k]})$ for $i \in [\ell]$ and $j \in [\ell_{\text{out}}]$
- sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e}_a \leftarrow \chi^m$, $\mathbf{e}_{i,j} \leftarrow \hat{\chi}$ for $i \in [\ell]$ and $j \in [\ell_{\text{out}}]$, and $e_{\text{msg}} \leftarrow \chi$
- set $\mathbf{c}_a = \mathbf{s}^\top \mathbf{A} + \mathbf{e}_a^\top$, $\mathbf{c}_{i,j}^{\text{prf}} = \mathbf{s}^\top (\mathbf{B}_{i,j}^{\text{prf}} + r_{i,j}^{\text{enc}} \cdot \mathbf{G}) + \mathbf{e}_{i,j}^\top$, and $c_{\text{msg}} = \mathbf{s}^\top \mathbf{u} + e_{\text{msg}} + \text{msg} \cdot \lceil \frac{q}{2} \rceil$ where $r_{i,j}^{\text{enc}}$ is the j -th bit of r_i^{enc}

Output $\text{ct}_y = (y, \mathbf{c}_a, \{\mathbf{c}_{i,j}^{\text{prf}}\}_{i \in [\ell], j \in [\ell_{\text{out}}]}, c_{\text{msg}}, \{r_i^{\text{enc}}\}_{i \in [\ell]})$

- $\text{Dec}(\text{pp}, \text{ct}_y, \text{sk}_x)$. On input the public parameters pp , a ciphertext ct_y and secret key sk_x do the following:

- parse $\text{ct}_y = (y, \mathbf{c}_a, \{\mathbf{c}_{i,j}^{\text{prf}}\}_{i \in [\ell_y], j \in [\ell_{\text{out}}]}, c_{\text{msg}}, \{r_i^{\text{enc}}\}_{i \in [\ell_y]})$
- parse $\text{sk}_x = (x, \mathbf{t}_{\mathbf{u},x}^{\text{comp}}, \{r_i\}_{i \in [\ell_x]})$
- parse $y \in \{0, 1\}^{\ell_y}$ and $x \in \{0, 1\}^{\ell_x}$ with $\ell_y, \ell_x \leq \ell_{\text{str}}$
- if x is not a prefix of y , then output $\perp$
- if x is a prefix of y , and there exists $i \in [\ell_x]$ such that $r_i^{\text{enc}} = r_i$, then output $\perp$
- set $\mathbf{c}_i^{\text{comp}} = \text{EvalCT}\left(f_{r_i}^{\text{eq}}, \{\mathbf{B}_{i,j}^{\text{prf}}, r_{i,j}^{\text{enc}}, \mathbf{c}_{i,j}^{\text{prf}}\}_{j \in [\ell_{\text{out}}]}\right)$ for $i \in [\ell_x]$
- set $\mathbf{F}_x = \mathbf{A} \parallel \mathbf{B}_{f_1^{\text{comp}}} \parallel \dots \parallel \mathbf{B}_{f_{\ell_x}^{\text{comp}}}$
- set $\mathbf{c}'_{\text{msg}} = (\mathbf{c}_a \parallel \mathbf{c}_1^{\text{comp}} \parallel \dots \parallel \mathbf{c}_{\ell_x}^{\text{comp}})^\top \cdot \mathbf{t}_{\mathbf{u},x}^{\text{comp}}$
- if $|c_{\text{msg}} - \mathbf{c}'_{\text{msg}}| \leq q/4$, then set $\text{msg} = 0$ else set $\text{msg} = 1$ (assuming the output is not $\perp$ until this step)

Output msg

4.2 Lattice Parameters

We provide an asymptotic parameter analysis and selection for our scheme. We give necessary details in this section, that are required for the correctness proof in the next section.

Let λ be a security parameter and $\ell_{\text{str}} = \ell_{\text{str}}(\lambda)$ be an upper bound on the string length that is polynomial in λ , and thus we can use only λ as our point of reference below. Let $d_{\text{ABE}} = d_{\text{ABE}}(\lambda)$ be an upper bound on the depth of $f_{\ell_{\text{str}}}^{\text{comp}}$. Let $d_{\text{eq}} = d_{\text{eq}}(\lambda)$ be an upper bound on the depth of f^{eq} ,

which can be made logarithmic in its input size $\ell_{\text{out}} = \ell_{\text{out}}(\lambda)$ (i.e. in NC^1). Let $d_{\text{PRF}} = d_{\text{PRF}}(\lambda)$ be an upper bound on the PRF evaluation circuit. Let $n = n(\lambda), m = m(\lambda), q = q(\lambda)$ be lattice parameters.

We first analyze the norm of the secret keys. This allows us to provide a bound such that Proposition 2.6 holds.

Simulated public keys. Using Lemma 2.7 for the simulated game, we get an upper bound on the largest singular value: $s_1 = s_1(\mathbf{R}_{f_{\ell_{\text{str}}}^{\text{comp}}}) \leq m \cdot \|\mathbf{R}_{f_{\ell_{\text{str}}}^{\text{comp}}}\|_{\infty} = m^{O(d_{\text{ABE}})}$ since $d_{\text{ABE}} = \omega(1)$.

Secret key bound. We follow the analysis from [CHKP10, MP12] for the basis sampling parameter σ_i , as required for a given $x \in \{0, 1\}^{\ell_x}$. We require for each $i \in [\ell_x]$ that,

$$\sigma_i \geq \sigma_0 \cdot \left(\sqrt{(i+1)m} \right)^i \cdot \left(\omega(\sqrt{\log(i \cdot m)}) \right)^{i+1}$$

where $\sigma_0 \geq s_1$. The above imply that **Keygen** samples from a discrete Gaussian with parameter σ_{ℓ_x} .

We now analyze the error magnitude for encryption and decryption.

Error magnitude. We first analyze the error magnitude in ct_y , and then in the resulting ciphertext after the ABE evaluation on ct_y that is required for decryption.

We sample the error $\mathbf{e}_a$ for $\mathbf{c}_a$ in ct_y , from $\mathcal{D}_{\mathbb{Z}^m, \sqrt{m}}$ and by Lemma 2.2 we get $\|\mathbf{e}_a\| \leq m$. Using Lemma 2.7 for the simulated game, we get an upper bound of $B = \|\mathbf{e}_a\| \cdot m \|\mathbf{R}_{f_{i,j}^{\text{prf}}}\|_{\infty} = m^{O(d_{\text{PRF}})}$ since $d_{\text{PRF}} = \omega(1)$. By Corollary 2.3 we sample $\mathbf{e}_{i,j}$ from $\mathcal{D}_{\mathbb{Z}^m, B'}$ with parameter $B' = (m^{O(d_{\text{PRF}})})^{\omega(1)}$. $m^{O(d_{\text{PRF}})}$ so that $\mathcal{D}_{\mathbb{Z}^m, B'}$ is swallowing for any B -bounded distribution (over the integers).

By Lemma 2.7 the error $\mathbf{e}_i^{\text{comp}}$ in $\mathbf{c}_i^{\text{comp}}$ is bounded by $\|\mathbf{e}_i^{\text{comp}}\|_{\infty} \leq B' \cdot m^{O(d_{\text{eq}})}$. To decrypt we invoke **SamPre** with σ_{ℓ_x} , as bounded above, to sample $\mathbf{t}_{\mathbf{u},x}^{\text{comp}}$. Thus, the error e'_{msg} in $\mathbf{c}'_{\text{msg}}$ is upper bounded by,

$$|e'_{\text{msg}}| \leq m \cdot \ell_x \cdot (\|\mathbf{e}_{\ell_x}^{\text{comp}}\|_{\infty} \cdot \|\mathbf{t}_{\mathbf{u},x}^{\text{comp}}\|_{\infty})$$

Concrete parameter selection. It is not hard to see that $m^{O(d_{\text{PRF}})}$ can be $m^{2d_{\text{PRF}}}$, and $m^{O(d_{\text{eq}})}$ can be $m^{\log(\ell_{\text{out}})}$, where we omit the multiplicative constant in the exponent of $m^{\log(\ell_{\text{out}})}$ to simplify our analysis. Given this, our analysis above implies,

$$s_1 \leq m^{2d_{\text{ABE}}+1}, \quad \sigma_{\ell_x} \geq m^{2d_{\text{ABE}}+1} \cdot m^{\ell_x/2} \cdot (\ell_x+1)^{\ell_x/2}, \quad \text{and thus we can choose } \sigma_{\ell_x} \leq m^{2d_{\text{ABE}}+\ell_x} \cdot (\ell_x)^{\ell_x}.$$

Therefore, $\sigma_{\ell_{\text{str}}} \leq m^{2d_{\text{ABE}}+\ell_{\text{str}}} \cdot (\ell_{\text{str}})^{\ell_{\text{str}}}$.

We then set $B' = (m^{2\log(\ell_{\text{out}})})^{\log(\lambda)+1} = m^{2\log(\ell_{\text{out}}) \cdot (\log(\lambda)+1)}$, and thus $\|\mathbf{e}_{\ell_x}^{\text{comp}}\|_{\infty} = B' \cdot m^{2\log(\ell_{\text{out}})}$, which implies $\|\mathbf{e}_{\ell_x}^{\text{comp}}\|_{\infty} \leq \ell_x \cdot m^{4\log(\ell_{\text{out}})\log(\lambda)}$. In addition $\|\mathbf{t}_{\mathbf{u},x}^{\text{comp}}\|_{\infty} \leq \sigma_{\ell_x}$, and thus:

$$|e'_{\text{msg}}| \leq (\ell_x)^{\ell_x} \cdot m^{\ell_x} \cdot m^{2d_{\text{ABE}}} \cdot m^{4\log(\ell_{\text{out}})\log(\lambda)}$$

We set $q = 5 \cdot |e'_{\text{msg}}|$ after we substitute $\ell_x = \ell_{\text{str}}$ in the above expression, and thus obtain decryption correctness for all $x \in \{0, 1\}^{\ell_{\text{str}}}$.

We set $n \geq \lambda$. We set $m = (n+1)\log q + \log^2(n)$. We set $\chi = \mathcal{D}_{\mathbb{Z}, \sqrt{m}}$. Set $\hat{\chi} = \mathcal{D}_{\mathbb{Z}^m, B'}$.

Lattice approximation factors. The quantity $n \cdot q / B'$ determines the approximation factor γ of the worst-case lattice problems. We can safely assume that ℓ_{str} is upper bounded by a polynomial in λ . Thus, we rely on the polynomial hardness of worst-case lattice problems with a sub-exponential factor γ that is upper bounded by

$$(\ell_{\text{str}})^{\ell_{\text{str}}} \cdot m^{\ell_{\text{str}}} \cdot m^{2d_{\text{ABE}}} \cdot m^{2\log(\ell_{\text{out}})\log(\lambda)} \leq 2^{\lambda^{\varepsilon}} \quad \text{for some } \varepsilon \in (0, 1/2).$$

4.3 Correctness

Lemma 4.2. The PRE construction in Section 4.1 satisfies the correctness property in Definition 3.1.

Proof. For all $\text{msg} \in \{0, 1\}$, all positive integers $\lambda, \ell_{\text{str}}$, all $x \in \{0, 1\}^{\ell_x}$ with $\ell_x \leq \ell_{\text{str}}$, and parameters generated by the Setup algorithm in Section 4.1 the following hold.

First, with probability $1 - \text{negl}(\lambda, \ell_{\text{str}})$ over the random choice of $k^{\text{real}}, k^{\text{fake}}, k^{\text{enc}}$ we have that $k^{\text{real}} \neq k^{\text{enc}}$ and $k^{\text{fake}} \neq k^{\text{enc}}$. By the statistical injectivity of the PRF family and the correctness of the homomorphic evaluation algorithms from Lemma 2.7, during decryption we obtain,

$$\mathbf{c}_a \parallel \mathbf{c}_1^{\text{comp}} \parallel \dots \parallel \mathbf{c}_{\ell_x}^{\text{comp}} = \mathbf{s}^\top \mathbf{F}_x + \mathbf{e}_x^\top$$

with $\mathbf{F}_x = \mathbf{A} \parallel \mathbf{B}_{f_1^{\text{comp}}} \parallel \dots \parallel \mathbf{B}_{f_{\ell_x}^{\text{comp}}}$.

By the correctness of SamPre in Lemma 2.4, when given as input $\mathbf{T}_A = \mathbf{T}_x$ where $\mathbf{T}_x$ is sampled using Lemma 2.5 and used in Keygen it holds that,

$$(\mathbf{s}^\top \mathbf{F}_x + \mathbf{e}_x^\top) \cdot \mathbf{t}_{\mathbf{u}, x}^{\text{comp}} = \mathbf{s}^\top \mathbf{u} + e'_{\text{msg}}$$

where $|e'_{\text{msg}}| \leq q/5$ as implied by the analysis and parameter selection in Section 4.2. Finally, since $|e_{\text{msg}}| \leq m$ and $q = m^{\omega(1)}$ it follows that,

$$|e'_{\text{msg}} - e_{\text{msg}}| \leq \frac{q}{4}$$

Thus, we obtain decryption correctness with probability $1 - \text{negl}(\lambda, \ell_{\text{str}})$. $\square$

4.4 Security Proof

Lemma 4.3. The PRE construction in Section 4.1 satisfies the security property in Definition 3.1.

Proof. We prove the lemma with the following sequence of hybrids.

$\mathcal{H}_0$: This is the real game under Definition 3.1 with the construction from Section 4.1.

$\mathcal{H}_1$: In this hybrid, we change how the public parameters pp are generated. We now sample $\mathbf{R}_i \xleftarrow{\$} \{-1, 1\}^{m \times m}$ and set

$$\text{pp}_{\text{sim}} = \{\mathbf{u}, \mathbf{A}, \mathbf{A}\mathbf{R}_1 - k_1^{\text{real}} \cdot \mathbf{G}, \dots, \mathbf{A}\mathbf{R}_{\ell_k} - k_{\ell_k}^{\text{real}} \cdot \mathbf{G}\} \quad \text{and} \quad \text{msk}_{\text{sim}} = \{\mathbf{R}_1, \dots, \mathbf{R}_{\ell_k}, k^{\text{real}}, k^{\text{fake}}\}$$

By the Leftover Hash Lemma from Lemma 2.1 and a Markov chain argument in [BV16, Section 4.2] we have that $\text{pp} \approx^{\$} \text{pp}_{\text{sim}}$ and thus $\mathcal{H}_0 \approx \mathcal{H}_1$.

$\mathcal{H}_2$: In this hybrid, we change how we answer the challenge ciphertext ct_{y^*} for $y^* \in \{0, 1\}^{\ell_{y^*}}$ and specifically how we sample k^{enc} .

We sample k^{enc} such that $k^{\text{enc}} = k^{\text{real}}$ which implies that $r_i^{\text{enc}} = r_i^{\text{real}}$ for all $i \in [\ell_{y^*}]$.

We show that this change goes unnoticed by the adversary, by arguing that the PRF values it obtains in ct_{y^*} using k^{real} do not appear in any requested secret key sk_x .

For every requested key sk_x with $x \in \{0, 1\}^{\ell_x}$, the adversary obtains for $i \in [\ell_x - 1]$ the PRF value $r_{x,i}^{\text{fake}} = f_{x,i}^{\text{prf}}(k^{\text{fake}})$, and for $i = \ell_x$ the PRF value $r_{x,\ell_x}^{\text{real}} = f_{x,\ell_x}^{\text{prf}}(k^{\text{real}})$.

For ct_{y^*} the adversary obtains for $i \in [\ell_{y^*}]$ the PRF value $r_{y^*,i}^{\text{real}} = f_{y^*,i}^{\text{prf}}(k^{\text{real}})$.

We proceed with three cases to show that the PRF values under k^{real} in every requested sk_x and in ct_{y^*} , do not intersect.

- If $\ell_x > \ell_{y^*}$ then the adversary obtains in $\mathbf{sk}_x$ the PRF values $r_{x,i}^{\text{fake}}$ for all $i \leq \ell_{y^*}$. These are different from $r_{y^*,i}^{\text{real}}$ in $\mathbf{ct}_{y^*}$ for all $i \in [\ell_{y^*}]$.
- If $\ell_x = \ell_{y^*}$ then the adversary obtains in $\mathbf{sk}_x$ the PRF values $r_{x,i}^{\text{fake}}$ for all $i < \ell_{y^*}$. These are different from $r_{y^*,i}^{\text{real}}$ in $\mathbf{ct}_{y^*}$ for all $i \in [\ell_{y^*} - 1]$. By Definition 3.1 it is not allowed to output $y^* = x$, and thus $r_{y^*,\ell_{y^*}}^{\text{real}} \neq r_{x,\ell_{y^*}}^{\text{real}}$.
- If $\ell_x < \ell_{y^*}$ then by Definition 3.1 for any $\mathbf{sk}_x$ the string x cannot be prefix of y^* , which implies that $x(1,i) \neq y^*(1,i)$ for all $i \in [\ell_x]$. This implies that $r_{x,\ell_x}^{\text{real}} \neq r_{y^*,\ell_x}^{\text{real}}$. Also, for $i \in [\ell_x - 1]$ we obtain $r_{x,i}^{\text{fake}}$ from $\mathbf{sk}_x$ and $r_{y^*,i}^{\text{real}}$ from $\mathbf{ct}_{y^*}$, and these values are all different since $k^{\text{real}} \neq k^{\text{fake}}$.

The above imply that the adversary obtains different PRF values under k^{real} when it is used both for $\mathbf{ct}_{y^*}$ and every requested $\mathbf{sk}_x$. Thus, under PRF pseudorandomness the change $k^{\text{enc}} = k^{\text{real}}$ goes unnoticed, and below we bound this reduction loss.

Assume the adversary $\mathcal{A}$ makes $Q = \text{poly}(\lambda)$ key queries in total. Let $\text{Adv}_{\mathcal{A}}^{\text{PRF}}(1^\lambda, 1^{\ell_{\text{str}}}) = \text{negl}(\lambda, \ell_{\text{str}})$ be the pseudorandomness loss for one PRF value. The total pseudorandomness reduction loss is at most $(Q + 1) \cdot \ell_{\text{str}} \cdot \text{Adv}_{\mathcal{A}}^{\text{PRF}}(1^\lambda, 1^{\ell_{\text{str}}})$, and remains $\text{negl}(\lambda, \ell_{\text{str}})$.

The above imply $\mathcal{H}_1 \stackrel{c}{\approx} \mathcal{H}_2$.

$\mathcal{H}_3$: In this hybrid, we switch from using the algorithm EvalPK to using EvalSimPK from Lemma 2.7. This is a syntactic change. For $\mathbf{ct}_{y^*}$ and $i \in [\ell_{y^*}]$, $j \in [\ell_{\text{out}}]$ we have $C_{i,j}(k^{\text{enc}}) = 0$. This implies that there is no $\mathbf{G}$ -trapdoor in $\mathbf{ct}_{y^*}$,

$$\mathbf{B}_{i,j}^{\text{prf}} = \mathbf{A}\mathbf{R}_{i,j}^{\text{prf}} - C_{i,j}(k^{\text{enc}}) \cdot \mathbf{G} = \mathbf{A}\mathbf{R}_{i,j}^{\text{prf}}$$

Following the last steps of the encryption algorithm, we get:

- sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e}_a \leftarrow \chi^m$, $\mathbf{e}_{i,j} \leftarrow \hat{\chi}$ for $i \in [d]$ and $j \in [\ell_{\text{out}}]$, and $e_{\text{msg}} \leftarrow \chi$
- set $\mathbf{c}_a = \mathbf{s}^\top \mathbf{A} + \mathbf{e}_a^\top$, $\mathbf{c}_{i,j}^{\text{prf}} = \mathbf{s}^\top \mathbf{A}\mathbf{R}_{i,j}^{\text{prf}} + \mathbf{e}_{i,j}^\top$, and $c_{\text{msg}} = \mathbf{s}^\top \mathbf{u} + e_{\text{msg}} + \text{msg} \cdot \lceil \frac{q}{2} \rceil$

$$\text{Output } \mathbf{ct}_{y^*} = \left(y^*, \mathbf{c}_a, \left\{ \mathbf{c}_{i,j}^{\text{prf}} \right\}_{i \in [\ell_{y^*}], j \in [\ell_{\text{out}}]}, c_{\text{msg}}, \left\{ r_i^{\text{enc}} \right\}_{i \in [\ell_{y^*}]} \right)$$

The above imply $\mathcal{H}_2 \equiv \mathcal{H}_3$.

$\mathcal{H}_4$: In this hybrid, we change again how we answer the challenge ciphertext $\mathbf{ct}_{y^*}$ and specifically how we create $\mathbf{c}_{i,j}^{\text{prf}}$. Using the algorithm EvalSimPK from Lemma 2.7 and msk_{sim} , we obtain $\mathbf{R}_{f_{i,j}^{\text{prf}}}$. We now compute

$$\mathbf{c}_{i,j}^{\text{prf}} = \mathbf{c}_a \cdot \mathbf{R}_{f_{i,j}^{\text{prf}}} + \mathbf{e}_{i,j}^\top = \mathbf{s}^\top \mathbf{A}\mathbf{R}_{f_{i,j}^{\text{prf}}} + \left(\mathbf{e}_a^\top \cdot \mathbf{R}_{f_{i,j}^{\text{prf}}} + \mathbf{e}_{i,j}^\top \right)$$

where $\mathbf{e}_{i,j}^\top \leftarrow \hat{\chi}$.

By the parameter selection for distributions $\hat{\chi}$ and χ from Section 4.2, and the upper bound on $\|\mathbf{R}_{f_{i,j}^{\text{prf}}}\|_\infty$ from the same section, it follows that $\mathbf{e}_{i,j}^\top$ is swallowed for $\mathbf{e}_a^\top \cdot \mathbf{R}_{f_{i,j}^{\text{prf}}}$. By Corollary 2.3 we get that the $\mathbf{c}_{i,j}^{\text{prf}}$ are statistically close the ones from the previous hybrid. Compared to

the previous hybrid though, in this hybrid we do not need to know $\mathbf{s}$ and $\mathbf{e}_a^\top$ to create the $\mathbf{c}_{i,j}^{\text{prf}}$.

The above imply $\mathcal{H}_3 \stackrel{s}{\approx} \mathcal{H}_4$.

$\mathcal{H}_5$: In this hybrid, we change how we answer secret key queries for a string $x \in \{0,1\}^{\ell_x}$, and specifically how we sample $\mathbf{T}_x$. From Lemma 2.7 combined with pp_{sim} it holds that, $\mathbf{B}_{f_i}^{\text{comp}} = \mathbf{A}\mathbf{R}_{f_i}^{\text{comp}}$ for $i \in [\ell_x - 1]$ and $\mathbf{B}_{f_{\ell_x}}^{\text{comp}} = \mathbf{A}\mathbf{R}_{f_{\ell_x}}^{\text{comp}} + \mathbf{G}$. That is,

$$\mathbf{A} \parallel \mathbf{B}_{f_1}^{\text{comp}} \parallel \cdots \parallel \mathbf{B}_{f_{\ell_x}}^{\text{comp}} = \mathbf{A} \parallel \mathbf{A}\mathbf{R}_{f_1}^{\text{comp}} \parallel \cdots \parallel \mathbf{A}\mathbf{R}_{f_{\ell_x}}^{\text{comp}} + \mathbf{G}$$

We can now use the `SampleBasisRight` algorithm from Lemma 2.5,

$$\mathbf{T}_x \leftarrow \text{SampleBasisRight}(\mathbf{A}, \mathbf{F}_x, \mathbf{R}_{f_{\ell_x}}^{\text{comp}}, \mathbf{G}, \sigma_{\ell_x})$$

where $\mathbf{F}_x = \mathbf{B}_{f_1}^{\text{comp}} \parallel \cdots \parallel \mathbf{B}_{f_{\ell_x}}^{\text{comp}}$. By Proposition 2.6, and Lemma 2.5, and our choice of parameter σ_{ℓ_x} in Section 4.2 we obtain $\mathcal{H}_4 \stackrel{s}{\approx} \mathcal{H}_5$.

$\mathcal{H}_6$: In this hybrid, we change how we sample the matrix $\mathbf{A}$. We now sample $\mathbf{A} \stackrel{s}{\leftarrow} \mathbb{Z}_q^{n \times m}$, instead of $(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \text{TrapGen}(1^n, 1^m, q)$.

By Lemma 2.4 it holds that $\mathcal{H}_5 \stackrel{s}{\approx} \mathcal{H}_6$.

$\mathcal{H}_7$: In this hybrid, we change again how we answer the challenge ciphertext ct_{y^*} . We output

$$\text{ct}_{y^*} = \left(y^*, \mathbf{c}_a, \left\{ \mathbf{c}_{i,j}^{\text{prf}} \right\}_{i \in [\ell_{y^*}], j \in [\ell_{\text{out}}]}, c_{\text{msg}}, \{r_i^{\text{enc}}\}_{i \in [\ell_{y^*}]} \right),$$

where $\mathbf{c}_a \stackrel{s}{\leftarrow} \mathbb{Z}_q^m$, $\mathbf{c}_{i,j}^{\text{prf}} \stackrel{s}{\leftarrow} \mathbb{Z}_q^m$ and $c_{\text{msg}} \stackrel{s}{\leftarrow} \mathbb{Z}_q$. Under the LWE assumption we have that

$$\mathcal{H}_6 \stackrel{c}{\approx} \mathcal{H}_7.$$

This implies that the challenge message $\text{msg} \stackrel{s}{\leftarrow} \{0,1\}$ is computationally hidden under the LWE assumption with parameters $(n, m, q, \hat{\chi})$ as defined in Section 4.2. Thus, for any PPT adversary $\mathcal{A}$ we have $\text{Adv}_{\mathcal{A}}^{\text{PRE}}(1^\lambda, 1^{\ell_{\text{str}}}) = \text{negl}(\lambda, \ell_{\text{str}})$ which proves Lemma 4.3. $\square$

Acknowledgements

We thank Daniel Wichs and the anonymous TCC 2021 reviewers for useful comments about adaptively secure HIBE, on a previous version of this manuscript.

References

- [ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Heidelberg, May / June 2010.
- [ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 98–115. Springer, Heidelberg, August 2010.
- [ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 657–677. Springer, Heidelberg, August 2015.
- [ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Heidelberg, August 2009.
- [AFS19] Prabhanjan Ananth, Xiong Fan, and Elaine Shi. Towards attribute-based encryption for RAMs from LWE: Sub-linear decryption, and more. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 112–141. Springer, Heidelberg, December 2019.
- [Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.
- [Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In Jiri Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *ICALP 99*, volume 1644 of *LNCS*, pages 1–9. Springer, Heidelberg, July 1999.
- [AP09] Joël Alwen and Chris Peikert. Generating Shorter Bases for Hard Random Lattices. In Susanne Albers and Jean-Yves Marion, editors, *STACS*, volume 3 of *LIPICs*, page 75–86. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.
- [BB04] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, Heidelberg, May 2004.
- [BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.
- [BGG⁺14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014.

- [BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, March 2014.
- [BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 410–428. Springer, Heidelberg, August 2013.
- [BLP⁺13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013.
- [BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 353–370. Springer, Heidelberg, August 2014.
- [BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Heidelberg, April 2012.
- [BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011.
- [BV16] Zvika Brakerski and Vinod Vaikuntanathan. Circuit-ABE from LWE: Unbounded attributes and semi-adaptive security. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 363–384. Springer, Heidelberg, August 2016.
- [BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 535–554. Springer, Heidelberg, February 2007.
- [BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013.
- [CGW15] Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system ABE in prime-order groups via predicate encodings. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 595–624. Springer, Heidelberg, April 2015.
- [CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 255–271. Springer, Heidelberg, May 2003.
- [CHK07] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. *Journal of Cryptology*, 20(3):265–294, July 2007.

- [CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, Heidelberg, May / June 2010.
- [CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 577–607. Springer, Heidelberg, August 2018.
- [CW13] Jie Chen and Hoeteck Wee. Fully, (almost) tightly secure IBE and dual system groups. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 435–460. Springer, Heidelberg, August 2013.
- [CW14a] Jie Chen and Hoeteck Wee. Dual system groups and its applications — compact HIBE and more. Cryptology ePrint Archive, Report 2014/265, 2014. <https://eprint.iacr.org/2014/265>.
- [CW14b] Jie Chen and Hoeteck Wee. Semi-adaptive attribute-based encryption and improved delegation for Boolean formula. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 277–297. Springer, Heidelberg, September 2014.
- [Dam90] Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *CRYPTO’89*, volume 435 of *LNCS*, pages 416–427. Springer, Heidelberg, August 1990.
- [DG17a] Nico Döttling and Sanjam Garg. From selective IBE to full IBE and selective HIBE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 372–408. Springer, Heidelberg, November 2017.
- [DG17b] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 537–569. Springer, Heidelberg, August 2017.
- [DGHM18] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 3–31. Springer, Heidelberg, March 2018.
- [DKN⁺20] Alex Davidson, Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively secure constrained pseudorandom functions in the standard model. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 559–589. Springer, Heidelberg, August 2020.
- [DKNY18] Alex Davidson, Shuichi Katsumata, Ryo Nishimaki, and Shota Yamada. Constrained PRFs for bit-fixing (and more) from OWFs with adaptive security and constant collusion resistance. Cryptology ePrint Archive, Report 2018/982, 2018. <https://eprint.iacr.org/2018/982>.
- [DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM journal on computing*, 38(1):97–139, 2008.

- [FKPR14] Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive security of constrained PRFs. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 82–101. Springer, Heidelberg, December 2014.
- [GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984.
- [GH09] Craig Gentry and Shai Halevi. Hierarchical identity based encryption with polynomially many levels. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 437–456. Springer, Heidelberg, March 2009.
- [GK VW20] Rishab Goyal, Venkata Koppula, Satyanarayana Vusirikala, and Brent Waters. On perfect correctness in (lockable) obfuscation. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 229–259. Springer, Heidelberg, November 2020.
- [GKW16] Rishab Goyal, Venkata Koppula, and Brent Waters. Semi-adaptive security and bundling functionalities made generic and easy. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 361–388. Springer, Heidelberg, October / November 2016.
- [GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In Chris Umans, editor, *58th FOCS*, pages 612–621. IEEE Computer Society Press, October 2017.
- [GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309.
- [GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.
- [GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 548–566. Springer, Heidelberg, December 2002.
- [GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.
- [GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 545–554. ACM Press, June 2013.

- [GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, Heidelberg, August 2015.
- [HKKW19] Dennis Hofheinz, Akshay Kamath, Venkata Koppula, and Brent Waters. Adaptively secure constrained pseudorandom functions. In Ian Goldberg and Tyler Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 357–376. Springer, Heidelberg, February 2019.
- [HL02] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 466–481. Springer, Heidelberg, April / May 2002.
- [JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability Obfuscation from Well-Founded Assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, page 60–73, New York, NY, USA, 2021.
- [Kim20] Sam Kim. Key-homomorphic pseudorandom functions from LWE with small modulus. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 576–607. Springer, Heidelberg, May 2020.
- [KL15] Lucas Kowalczyk and Allison Bishop Lewko. Bilinear entropy expansion from the decisional linear assumption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 524–541. Springer, Heidelberg, August 2015.
- [KNYY20] Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively secure inner product encryption from LWE. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 375–404. Springer, Heidelberg, December 2020.
- [KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 669–684. ACM Press, November 2013.
- [KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, Heidelberg, April 2008.
- [KW19] Lucas Kowalczyk and Hoeteck Wee. Compact adaptively secure ABE for NC^1 from k -Lin. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 3–33. Springer, Heidelberg, May 2019.
- [Lew12] Allison B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 318–335. Springer, Heidelberg, April 2012.

- [LOS⁺10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, Heidelberg, May / June 2010.
- [LW10] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 455–479. Springer, Heidelberg, February 2010.
- [LW11] Allison B. Lewko and Brent Waters. Unbounded HIBE and attribute-based encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 547–567. Springer, Heidelberg, May 2011.
- [LW12] Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 180–198. Springer, Heidelberg, August 2012.
- [LW14] Allison B. Lewko and Brent Waters. Why proving HIBE systems secure is difficult. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 58–76. Springer, Heidelberg, May 2014.
- [LW15] Vadim Lyubashevsky and Daniel Wichs. Simple lattice trapdoor sampling from a broad class of distributions. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 716–730. Springer, Heidelberg, March / April 2015.
- [Mer90] Ralph C. Merkle. One way hash functions and DES. In Gilles Brassard, editor, *CRYPTO’89*, volume 435 of *LNCS*, pages 428–446. Springer, Heidelberg, August 1990.
- [MM11] Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 465–484. Springer, Heidelberg, August 2011.
- [MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012.
- [MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 21–39. Springer, Heidelberg, August 2013.
- [MR07] Daniele Micciancio and Oded Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM J. Comput.*, 37(1):267–302, 2007.
- [O’N10] Adam O’Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. <https://eprint.iacr.org/2010/556>.
- [OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 191–208. Springer, Heidelberg, August 2010.

- [Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 333–342. ACM Press, May / June 2009.
- [Pei16] Chris Peikert. A Decade of Lattice Cryptography. *Foundations and Trends in Theoretical Computer Science*, 10(4):283–424, 2016.
- [PRS17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th ACM STOC*, pages 461–473. ACM Press, June 2017.
- [Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.
- [Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.
- [Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO’84*, volume 196 of *LNCS*, pages 47–53. Springer, Heidelberg, August 1984.
- [SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.
- [SW08] Elaine Shi and Brent Waters. Delegating capabilities in predicate encryption systems. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfssdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 560–578. Springer, Heidelberg, July 2008.
- [Tsa19] Rotem Tsabary. Fully secure attribute-based encryption for t-CNF from LWE. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 62–85. Springer, Heidelberg, August 2019.
- [Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Heidelberg, August 2009.
- [Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 678–697. Springer, Heidelberg, August 2015.
- [WTSB16] David J. Wu, Ankur Taly, Asim Shankar, and Dan Boneh. Privacy, discovery, and authentication for the internet of things. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *ESORICS 2016, Part II*, volume 9879 of *LNCS*, pages 301–319. Springer, Heidelberg, September 2016.
- [WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In Chris Umans, editor, *58th FOCS*, pages 600–611. IEEE Computer Society Press, October 2017.